51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2013年06月
总第027期

程序员

专访原阿里系统工程师周鹏：转行卖水果的思维方式



专访Tom：GenMyModel在线UML建模工具背后的敏锐团队

GenMyModel的CTO Tom给网友们分享了新型在线UML建模工具的一些优势以及未来发展趋势的一些观点，同时也分享了他们背后敏锐的团队历时九个月研发首个beta版本的新型在线UML建模工具。

编程排行 Billboard

03 2013年6月编程语言排行榜：JavaScript重回前十

专题报道 Special report

06 专访：原阿里系统工程师周鹏转行卖水果的思维方式
09 心之所至 境由心生:架构师徐佳转行卖水果历程
11 轮椅上的风景：身残志坚程序员张肖敏北上寻梦

技术热点 Techlogy hot

15 IT民工创业之殇
19 专访Tom：GenMyModel在线UML建模工具背后的敏锐团队
22 将死的平台：.Net
23 编程思想：我现在是这样编程的
26 从100PV到1亿级PV网站架构演变
29 在Google做实习生是什么样的
31 从面向对象的设计模式看软件设计
33 大话科技：产品经理/码农/设计师的3P战争
35 如何留住你最棒的程序员
37 十年经历：软件开发的“三重门”
39 偷了世界的程序员
41 免费游戏设计法则：系列简介
43 我们“看不到”的残疾人程序员
45 “保卫妞妞”一位WEB工程师的手游初创之路
49 VS 2013在ALM上的改进
51 看曹政如何减少SQL请求
52 如何在网页中应用HTML 5音频
53 大数据与商业分析：新的TOP-DOWN纠结

■ 编者按

TIOBE编程语言排行榜6月榜出炉。C#跌出前五，他的位置由PHP顶替，这也是今年以来PHP首次闯入前五。值得注意的是JavaScript本月排名第十位， 重新进入排行榜前十。

2013年6月编程语言排行榜：JavaScript重回前十

【51CTO独家特稿】TIOBE编程语言排行榜6月榜出炉。本期排行榜前四位没有变化，C语言依旧稳居第一宝座，Java和Objective-C列位第二、第三，C++排在第四。C#跌出前五，他的位置由PHP顶替，这也是今年以来PHP首次闯入前五。

值得注意的是，JavaScript本月排名第十位， 重新进入排行榜前十。JavaScript是一种基于对象和事件驱动并具有相对安全性的客户端脚本语言，是时下最为流行的Web脚本语言。因此这样一种无处不在的语言迟迟没有进入前五，让人觉得无法理解。

现如今JavaScript正在积极扩大其应用领域，Node.js是一套基于JavaScript的工具包，以此用来编写高性能网络服务器，使JavaScript在服务器端得以应用。此外，基于Javascript开发的游戏（大部分基于浏览器）也在不断增加。

而其他语言在以JavaScipt为目标而诞生，比如CoffeeScript（排名139）， 该语言增强了JavaScript 的精简性与可读性，可被编译成高效的JavaScript；TypeScript （排名207）是JavaScript的一个超集，TypeScript用来为大型应用服务，而编译时它生成JavaScript 以确保兼容性。

2012年12月编程语言排行榜榜单

Position Jun 2013	Position Jun 2012	Delta in Position	Programming Language	Ratings Jun 2013	Delta Jun 2012
1	1	=	C	17.839%	+0.08%
2	2	=	Java	16.656%	+0.39%
3	4	↑	Objective-C	10.356%	+1.26%
4	3	↓	C++	8.819%	-0.64%
5	7	↑↑	PHP	5.987%	+0.70%
6	5	↓	C#	5.783%	-1.24%
7	6	↓	(Visual) Basic	4.348%	-1.70%
8	8	=	Python	4.183%	+0.33%
9	9	=	Perl	2.273%	+0.05%
10	11	↑	JavaScript	1.654%	+0.18%
11	10	↓	Ruby	1.479%	-0.20%
12	12	=	Visual Basic .NET	1.067%	-0.15%
13	17	↑↑↑↑	Transact-SQL	0.913%	+0.21%
14	14	=	Lisp	0.879%	-0.11%
15	16	↑	Pascal	0.779%	-0.07%
16	21	↑↑↑↑↑	Bash	0.711%	+0.09%
17	19	↑↑	PL/SQL	0.657%	+0.02%
18	13	↓↓↓↓↓	Delphi/Object Pascal	0.602%	-0.55%
19	18	↓	Ada	0.576%	-0.11%
20	22	↑↑	MATLAB	0.563%	0.00%

还原JavaScript的真实历史：

在许多资料，JavaScript的语源被追溯到一种名为Cmm的语言。同样是在这一溯源的过程中，人们发现“其实”JavaScript不是第一种网页中的脚本语言。现在一些所谓“公认”的情况是这样的：

大概在1992年，一家称作Nombas的公司开始开发一种叫做C减减（C-minus-minus，简称Cmm）的嵌入式脚本语言。这个脚本语言捆绑在一个叫做CEnvi的共享软件产品中，当Netscape Navigator崭露头角时，Nombas开发了一个可以嵌入网页中的CEnvi的版本。这些早期的试验称为EspressoPage（浓咖啡般的页面），它们代表了第一个在万维网上使用的客户端脚本语言。而Nombas丝毫没有料到它的理念将会成为因特网的一块重要基石。

然而，这是真实的情况吗？运行在Netscape中的第一个“客户端脚本语言”真的是Cmm的浓咖啡？又或者真的象Wiki中记述的那样，Brendan Eich在JavaScript引用了Cmm语言特性？不是的。尽管上述的资料看起来出权威：《Javascript高级程序设计》（Professional JavaScriptfor Web Developers），但他的确错了。

JavaScript与Cmm在语言特性上无关

为了弄明白JavaScript与Cmm的关系，我大概用了三天的时间，从网上收集了多个Cmm的版本。这些版本既包括早期的Cmm(1993年)，也包括在JavaScript初创时的Cmm(1995年)，还包括在后来，Cmm更名为ScriptEase的第一个版本(3.0, 1997年)，作为参考，我还考察了它在服务器端的版本。

遗憾的是，在1996年之前，在Cmm的2.x的最后一个版本之前，Cmm都并不是一个面向对象（或基对象）的语言，Cmm正如它自己的名字所说的一样：是一个精减版的C，而不是C++或以C++为基础的变种。所以Cmm中有“结构”，也有#include等，整个的体系是参考C语言的。

JavaScript最初的基本设计是怎样的？

我需要补充一下JavaScript初始设计目标。Brendan Eich在1995年4月前后被Netscape公司雇佣，目标是完成一套语言系统。最初的设计里，该语言系统是为Netscape的LiveWire战略服务的。该

战略彰显了Netscape当年的勃勃野心，它是Netscape公司的一个通用的Web开发环境，包括NetscapeEnterprise、FastTrack Server等。LiveWire架构也成为所有Web服务器提供SP(Server Page)技术的蓝本。例如在IIS中的ASP，以及更早期的IDC(Internet Database Connect)。

这种技术在服务器端通过内嵌于网页的LiveScript代码，使用名为database、DbPool、Cursor等的一组对象来存取LiveWire Database。作为整套的解决方案，Netscape在客户端网页上也提供

LiveScript脚本语言的支持，除了访问Array、String等这些内置对象之外，也可以访问window等浏览器对象。

不过并不清楚的是：LiveScript最初的设计是先考虑服务器端应用，还是先考虑网页中应用。但这些应用环境的决策上的变化，时间仅仅限制在1995年4月至1995年10月之前，因为当月发布的netscape 2.0 beta1已经包含了脚本支持。

在netscape 2.0 beta1中并不支持<script>标签，而只是在form表单元素中支持了onclick这类的事件。这时的脚本代码是用在HTML标签属性上的，也就是类似于：

```
<input type="button" onclick="alert('hello, world!')"> ■
```

本文未完，请阅读原文：
http://developer.51cto.com/art/201306/397895.htm

51CTO.com
技术成就梦想

开发月刊





出版方：

北京无忧创想信息技术有限公司

责任编辑：

彭凡　林师授　陈四芳

封面设计：

苍旭

联系方法：

邮箱：linss@51cto.com

电话：010-68476606-8123

出版日期：2012年12月21日

欢迎来稿，请发送邮件至

linss@51cto.com

说起程序员这个高薪职业其实都是很多人的梦想，当你进入职场以后就会发现原来程序员其实没有想象中的那么美好。在拿高薪同时伴随着强大的工作压力，作息时间不规律，人际沟通变少，慢慢的随着年纪的增长，一直处于超负荷运转的程序员应该何去何从？

其实程序员就是属于年轻人的职业， 随着年龄的增长在技术更新的大潮中便面临竞争力衰减的残酷现实，需要重新定位和转型。

以其在编程的队伍里面打拼，不如重新审核自己的职业规划寻找新的目标。

本期开发月刊将给大家分享一些转型和创业的原创，以及相关技术内容。

祝好！

51CTO开发频道寄语

专访：原阿里系统工程师周鹏转行卖水果的思维方式

□ 小林/文

可以说程序员是IT行业的重要组成的地基。在这行业要面临着一个很大的工作强度，以及技术更新换代快，作息时间不规律等问题。随着年龄的增长，程序员该何去何从，该怎样规划自己的职业，做好职业地转型是程序员必须面对的一个问题。

前段时间，微博上出现了部分IT人员转型卖水果的事情确实有着不小的轰动，各大媒体也有进行报道转载，为什么会出现这种情况呢？用周鹏自己的话说，在大部分的人眼里觉得水果这行业是一个贱业。从一个不愁吃穿的阿里巴巴高级运维转型到做一个人们眼中的贱业，确实让人困惑和质疑。

为解疑惑，51CTO记者专访了原阿里巴巴集团运维部系统工程师 @周鹏对自己转型的内心想法以及发展路线，分享给为打算转型的程序员提供参考。



原阿里巴巴集团运维部系统工程师 周 鹏

以下是相关采访内容：

记者：请问您是从什么时候开始转行卖水果？主要是基于一些什么样的原因？

周鹏：从去年六七月份开始想转行这个事情。我辞职是在去年四五月份，出来以后，开始也想了一些别的东西，不过后来才想到做水果。八月份的时候，实验性地做了一批龙眼，那时候公司还没有成立，我自己搞了一下。相当于是自己了解一下这个过程，不是盲目地走进去。

记者：您如何去看待目前这个职业？

周鹏：我觉得还行，应该算是一个理想了。就是你刚才说基于什么样原因，原因是很多的。首先是个人的成长，我之前在博客里面也写到，作为一个程序员说的话，我不是做程序员，我之前是做系统管理与运维方面的。我觉得只是做技术化，我们的视野比较窄，看到的东西和接触到的人比较少，因为你往往只会盯着一块。另外一个方面，也是想做点事情，一开始是有这个抱负。像易中天在《品三国》里面说的两个老大，他说曹操是有理想的，刘备是有抱负，一开始都想做点事情。我最开始做技术的时候是这样想的，在20岁那时候我刚刚开始接触计算机，对那个东西比较感兴趣。那时候有着很多的想法，但还是想在专业上有很深的造诣。这个发展我分为三个级别，最低的是专家，至少到专

家级别，再往上做到大师的级别，最后才能做到宗师的级别。

以前的想法是这样的，但是后来自己看的东西多了，接触的东西也多了，想问题的方式不一样，开始对金融、投资这些东西感兴趣。不过骨子里还是想做事，后来想做这个事情，变成一种理想，心里也就有一个蓝图。

记者：那么您有这样的想法决定要转型时，有没有遇到一些阻力？比如说像亲人的反对，还有朋友的一些不理解，你怎么去处理这些事情？

周鹏：阻力肯定是有了，我家里还好，我父母的话，从小各种事情都是我自己处理的，包括升学、填志愿都是我自己。但是作为朋友肯定有鼓励的，当然也有反对的。

其实我觉得，别人的意见我都会参考一下，但是不能被这些意见左右。虽然是出于好意，但是有些意见对你也是一种拖累，因为他自己也没做过这个事情。他提出的一些想法，你做一个参考，你自己去多思考一下。如果你觉得他说的有道理，你去想一想，你去改进。如果你觉得没道理的话，你想想为什么，就是这样的。反正就是这么一个意思，我也不太会说。

记者：就是不理解你为什么转这个行业？

周鹏：有些人觉得水果这个东西是一个贱业，很低贱的。可能不会说出来，但是潜意识里面会有这种感觉。也有朋友直接跟我说，我就不看好你，我觉得你不是做生意的料。像我们这种做技术出来的，有可能不那么八面玲珑，能说会道，而我肯定不是这样的人。我觉得在水果这个传统的行业最重要的还是以诚待人，真诚地去对待别人，不管是对合作还是对顾客，这是我的想法。就算我能说会道，但是我也不认为它仅仅是个生意，我觉得这是一个事业。如果只是一个生意的话，我这就有一个很简单的例子，我这次做樱桃，现在像这边的话，它这边已经出来了。如果只是做生意的话，我现在就可以发，现在就应该发，发的话，我就可以多做一批，这样就相当于我可以多做一批，但是为什么我没发？我在那边待了两天，就是盯着他们，看他们怎么销，但是在我了解的整个过程中他们的控制是比较粗，包括采家，包括他们的包装，这些东西不是很严格。所以无论是采摘，包装等一些细节方面，我们都会控制好，就像在IT行业一样，只要稍微不注重细节，就会导致很多意想不到的灾难。

记者：是的，从上面你所说的很多我感觉区别IT行业与传统行业不同的处理方式，那么IT转水果和传统水果行业有没有什么样的借鉴？

周鹏：借鉴的话肯定是有，因为我们做技术、做IT出来的，我们会带有一些色彩。从长远来说，我的规划是我们需要把农业这个行业，做到信息化、标准化，同时我们会有一个更严格的一个指标，或者流程控制。

还有就是我今天还在写关于运输过程中的一些压力测试，以及考虑的各种条件。针对各种条件我们都做一些测试，把它形成文本。包括我们推进的时候，目前执行的一个指标，就是一个品质的指标。我们会出一个核验单，产品的核验单，因为我们有自己的一套指标。对于指标，现在都是在按版本走，如果现在执行一个1.0的指标，往后会更严格，对各种东西更严格，我们会执行一个更高的指标，因为我之前做运维，主要强调可用性，还有用户体验，我们可能比传统的模式更加注重这一块。包括团队合作，不完全像传统那样，当然传统的也会有团队合作，而现在我们是从互联网一个角度来说。

记者：可能大家认为互联网团队之间更加密切一切。

周鹏：对。更为平等的氛围，这样的方式，更有创造力，更有创意的一些工作方式。

记者：就像你说的卖水果是否用上程序员工作中的一些思维模式和职业技能？

周鹏：职业技能，当然不是说卖水果，主要是因为我们有淘宝，后面还会有网站建设，肯定会用到一些这样的技能。我自己因为做运维，有些小事情自己处理也很快，反正就是一些小问题。以后团队建立强大起来，我就不会专门去做这件事情。对于模式方面会更严格。首先要严谨，这是最重要的。以前我们老大跟我们说，你们就是整个技术团队的守门员。我们就是守门员，我们考虑问题可能会模块化一点。

我自己的一个想法是我们现在本质上也是在构建一个系统，我们想把供应链这个东西都组合起来，实际上也是在整合一个系统。不过这个系统里面的模块不是代码，而是人，当然更复杂了，但是他本质上也是一个系统。就是这样，因为模式上可能会有一些可以借鉴的地方。

记者：恩，是这样的。那目前您的工作状态是什么样的？是否还像上班时候的一个朝九晚五这样的一个状态？

周鹏：那肯定不是，以前也不完全是朝九晚五，有任务的话肯定要半夜处理一些故障或者什么样的，这些以前也会有。现在的话也不是，我们现在资源比较匮乏，要多艰辛。做水果这个东西就是很艰辛，因为早上七点多钟爬起来去批发市场，一个一个批发市场去看行情什么东西的，这个很正常。

记者：呵呵，其实很多人也许会像我这么认为个人创业时间会自由些。不过转型管理层也是很多IT人士的方向，未来有没有打算返回IT行业这块？

周鹏：我个人觉得最重要的不是说做什么职位，而是说自己一个能力，还有一个认识水平的提升，以及看问题的一个视野。如果说将来自己能力提升了，不管最后创业结果怎么样，最起码一点，我看问题的方式，我做事情的能力有一定的提升了。不管我将来去干什么，都是有好处的。

比如技术这个东西你看着好像很有必然，但实际上多花点时间，大家都能学会。但是你处理问题的经验、你的视野、还有思考问题的方式、你的态度、心态这些东西，这个不是说多看看书，或者多操作就能够提高。我现在的想法就是让自己将来路子很宽，不管以后做什么。我在博客里面写了，在大峡谷，还有荒野里，视野很开阔，不管你将来怎么样，心态上是没有问题。

记者：我感觉现在的互联网发展，人人都离不开IT。就拿做水果这一行，以后如果有可能做大了以后，咱们后面也有一个运维模式，你要做网站建设推广之类的，也要用到一些IT的技能。以后会不会专注这一块，采摘水果交给别人去做？

周鹏：这样的，因为我现在在做这个事情，其实我就是比较看重领导力这些方面的能力，但以后这一块，其实在各个方面都会有比我强的人。我们去年讨论过，也跟朋友讨论过，作为一个leader和manager是有区别的。manager下边的人可能你会不如他。但是我觉得作为一个leader，很多人比你强。■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201306/398350.htm
点击进入周鹏淘宝店，也希望大家支持他。

心之所至 境由心生:架构师徐佳转行卖水果历程

■ 随着年龄的增长在技术更新的大潮中便面临竞争力衰减的残酷现实，需要重新定位和转型。以其在编程的队伍里面打拼，不如重新审核自己的职业规划寻找新的目标……

说起程序员这个高薪职业其实都是很多人的梦想，当你进入职场以后就会发现原来程序员其实没有想象中的那么美好。在拿高薪同时伴随着强大的工作压力，作息时间不规律，人际沟通变少，慢慢的随着年纪的增长，一直处于超负荷运转的程序员应该何去何从?

其实程序员就是属于年轻人的职业， 随着年龄的增长在技术更新的大潮中便面临竞争力衰减的残酷现实，需要重新定位和转型。以其在编程的队伍里面打拼，不如重新审核自己的职业规划寻找新的目标。

前新浪网架构师徐佳的转型，让我们更加明白了“心之所至，境由心生”的道理。51CTO记者就转型的问题跟徐佳电话聊了一些关于自己定位以及转型后的职业生涯。另外，也希望大家去徐佳的淘宝店支持一下。



2011

以下是相关的采访整理

51CTO：请问您从什么时候开始转行销售水果这行业，主要是因为一些什么样的原因?

徐佳：其实可以这么说，第一是在北京也漂了几年，挺想回老家。第二个就是说感觉上班，两点一线的生活比较枯燥，还是想扩大自己的视野。

51CTO：是否朝九晚五这样的一个状态，感觉比较枯燥?

徐佳：对。第三个就是说，自己的一个目标，包括做程序员、在做这行也到一定阶段了，自己原来预想的期望也达到了。

51CTO：就是说想尝试一下新的职业是吗?

徐佳：对，尝试一下新的生活方式。

51CTO：你在转型的过程中，有没有遇到一些相关的阻力，比如家人反对，朋友的不理解?

徐佳：对，都会遇到。他们认为你有这么高工资，出来以后还要重头开始，更没有好的待遇，工资也没了，很多人建议不要出来。自己既然已经决定出来了怎么办？所以只能说服了他们。

51CTO：有一些非常反对的一些事例，你怎么去处理这些事情?

徐佳：这个倒是没有，因为家里面的人，他们最多只是给建议，因为现在这么大了，自己能拿主

意。不管他们是否反对，对于我来说，都是由我自己决定。另外一个是说，他们想的就是说，你如果出来了，到时候没钱了怎么办？我觉得这倒不是问题，你必须要出来，自己要生存。我自己也存了一点钱，出来之后，比如说一年半载的不工作也有自己的生活保障，像这也是一个冒险的事情，思想上也会有些斗争。

51CTO：你从事水果行业也有半年多了是吧？

徐佳：对。之前我朋友要做，帮朋友做，后来自己也开始正式做这个事情。

51CTO：从你个人角度来看，从事水果行业和IT行业有没有什么可以借鉴的？

徐佳：肯定是有很大帮助的，相对来说是做项目，做项目从基础分析调研，还有设计，到了具体的实施，到后期的封装和找bug。用在水果上面来说用户的反馈，这样一个，流程是相通的。

51CTO：程序员在卖水果的方面，他在工作中的一些思维模式和职业技能可以用上，比如说淘宝店的设计管理、团队管理，还有自动化记账、电子商务什么的，这些你们都能用上吗？

徐佳：对，通过自己的技术入门比较快，可以通过自己一些技术方面的东西比别人更有优势。

51CTO：我之前也看到，因为网上传了你一张照片也是蛮多的，从事水果行业后有一些变化，目前对这个职业满意吗？

徐佳：也不是太满意，毕竟水果这方面比较难做，现在只是说自己的一种尝试。

51CTO：你现在做的这个工作，有像IT行业一样朝九晚五这样的一个状态吗？还是时间比较自由些？

徐佳：主要是自己思考的多一些，其他方面，时间都是比较紧，也比较累，因为很多事情必须想好之后，安排下去以后，自己就可以做点自己的事情。

51CTO：把事情先安排完，主要是自己的一些想法比较多是吗？自个儿也属于一个创业的阶段，不受别的一些什么之类的限制，不一定有限制，自己也就比较轻松一些，是这样的状态吗？

徐佳：对，心态比较轻松一些。

51CTO：程序员转型，大部分都是往管理层这一块转过去，你有没有打算返回IT行业的一个考虑？

徐佳：这个可能是说没多大打算，出来之后，就不走回头路了。以后比如说为了生计，或者是一年半载生意做亏了之后，或者为了求生之类的，也有可能。

51CTO：恩。那么以你的想法，程序员卖水果的思维方式和传统卖水果的有没有什么不同的？

徐佳：可能有些事情想的更周密一些，其他的就是说技术方面的优势，另外一个是人本身的思维方式，因为这个东西跟职业没有多大关系，如果你真的做生意的话，别人如果没读过书的话，可能他做生意就会，其实跟职业没多大关系，而且干一行，跟这行业的经验有很大关系。相当于你，肯定比别人是看了很多年的，相当于你从零起步。一手拿着小本子去学习，去做一些新的东西，新的知识，才能尽快做好。不是说在互联网这块有一定的优势，有了IT基础，在很多的传统行业你都能应该都有一定的优势。■

本文未完，请阅读原文：
http://developer.51cto.com/art/201306/399645.htm

轮椅上的风景：身残志坚 程序员张肖敏北上寻梦

□ 小林/文

“活着的人最重要的是要有灵魂、要有思想，最重要的是要有一颗感恩的心。只要一个人拥有这些，他便不是无用之人。一个人拥有这些，他便不是无用之人。一个人拥有这些，便是拥有的，是快乐的，是幸福的……”

---张海迪

20岁在大一学习道路桥梁检测专业的张肖敏本该是意气风发的年华，可是一场结膜性脑膜炎夺去了他的双腿。但是他始终微笑着面对不幸，凭着坚强的意志与执着，边开淘宝店边自考30多门课程，自学多种计算机编程语言，边吃着强效止疼药，边独自出行求学工作，为我们唱响了不屈的生命之歌。



图为 张肖敏在北京的照片

以下是51CTO记者（5月30号）对张肖敏采访的一些整理：

厄运的开始

张肖敏是在04年6月底上大一的时候发病，这个疾病发展的很快，是结核性脑膜炎，持续了差不多10天，看了一次没诊断出来，第二次去看的时候就影响到腿了。这个病很像感冒，一下子很难察觉出来，张肖敏跟我们说，需要有经验的医生才能够意识到，当时去的那医院没反应过来，他第二次去的时候就已经很严重了。

张肖敏是家里的独生子，他父亲是县供电有限公司的一名普通职工，母亲在他生病后就一直照顾着他，直到2012年初，此时张肖敏已经出来工作一年半了，他觉得自己应该要学会独立生活了，父母不可能陪着他一辈子，他也希望妈妈能够有她自己的生活，自己心里面的愧疚感也会少一些，就跟他妈妈商量，一开始家里人也挺担心的，但是妈妈回去以后他自己一直都照顾很好，也就逐渐的放心了，张肖敏家离福州市大概有100多公里，4个小时的车程，母亲偶尔来福州看一下，平时就自己在福州生活。

在张肖敏的空间里曾看到这样的一段话描述了他的母亲：

“在一听说我生病要住院，妈妈就给老家的阿姨和外婆打电话，但是只是一直在哭泣，都说不出话来，深刻的爱却被距离给阻隔了，妈妈的心里流

下了无奈的泪水。在火车上，妈妈总是趴在桌子上，心理默默的哭泣，阿姨在边上一直安慰，希望她能够吃点东西，妈妈沮丧的脸望着窗外的田野和田间三三两两的行人这一幅和谐的场面，心理更加的恐惧与不安。”

张肖敏日常的生活都是使用轮椅，上肢是完全正常的，这么多年已经习惯了，无论是吃饭、洗脸、刷牙、洗澡都能够解决，不过洗衣服、整理房间确实很不方便。在福州时每个星期固定叫钟点工来做下家务，洗衣服、洗床单，整理东西，平时没什么事，像吃饭叫餐，有时候也出去吃，基本是这样。

面对现实

在最初的一年里张肖敏并没有意识到病情的严重性，当时还想着最多休一年学就回去继续上学。在一年的治疗之后，张肖敏已经意识到他以后都将无法行走，只能依靠轮椅来生活，思想上特别的绝望，对未来很迷茫，不知道他自己能干什么，也有过一些轻生的想法，基本上就这么个状态，度日如年苟延残喘的活着。由于昂贵的医药费，当时断断续续的还在治疗，偶尔去医院，住院也住一小段时间，来回这样折腾，其他的时间上网，除了治疗之外什么都没干。直到06年1月他开始在淘宝经营网店，当时张肖敏母亲在南京还有一家鞋店，所以就有货源。

从这个网店开始他的生活状态有了一个转折性的变化，除了治疗以外，其它时间都用来经营这个网店，一方面生活特别的充实，思想上就变得简单了，没空去思考未来前途，另一方面网店的收入也让他感受到了一种成就感，其实自己能做一些事情，虽然不太多，每个月差不多一千的利润，但是对他心里的激励是巨大的。这个网店差不多做了14个月，有345个信用，百分百好评，一颗钻石。但是面对高昂的治疗费用，显然是杯水车薪。后面他妈妈把实体店转让了，一家人就一起回到老家福州了，没有条件再经营了，他自己也想把时间花在自学考试上，所以忍痛放弃了网店的经营。

聊到与其他程序员的区别在哪里时，张肖敏说这也是他这么多年一直在思考的问题，从广义上来说当程序员跟下肢残疾完全没什么关系，但是从狭义上来说，是对工作环境有需求的，需要有无障碍的环境能够方便轮椅通行。不过也正是因为行动不方便和坎坷的经历也历练了他工作吃苦耐劳，做事认真责任，为人稳重成熟的风格。所以在客观条件允许的情况下，自己就能扬长避短化劣势为优势，跟正常的程序员相比自己就是有竞争力的，也是后面选工作时首先考虑的一个因素，工作的环境一定是需要无障碍的。

自学之路

张肖敏在做淘宝的时候，一有时间 特别想学点西，首先想到的是学计算机，他觉得这对于以后的工作会有点关系，然后就去学了。同时也很清楚记得是07年5月1号，国家发布了残疾人就业条例，然后那个时候张肖敏开始意识到可能他自己以后考完本科就有可能去工作，但是具体能做什么其实他并不清楚，直到2010年初把程序的一些基本课程学完，开始对软件有些概念的时候，那个时候就开始接触PHP，然后在这个过程中强烈的意识到PHP就是以后自己的工作方向。

面对轮椅人生，张肖敏始终没有放弃过追求。

由于家住偏远山区，去一趟福州有100多公里，外出成了大问题。他家里人对他继续参加自考的决定难以理解，认为一个残疾人即便拿到再多的文凭最后还得靠别人照顾。面对精神压力与自考本身的

难度及费用，张肖敏依然顽强地面对，平日里除了康复锻炼外，他把所有时间都用在自学上。他家里人开始的时候不太支持，也许是对计算机软件没有太深的概念，到后面能够出去工作了，用电脑做PHP开发能够自食其力，也就放心了，没有什么太多要求。后面越做越好，对肖敏有了信心，也不担心什么了。

张肖敏从自学考试里面的基础课程到PHP开发都是自学的，因为有了C和C++的基础，学PHP相对来说容易些。后面的深入学习属于完全自学，等于驱动式的自学。张肖敏的自学能力本身在自学考试过程中已经锻炼出来了，基本上都是自己在家学，每回考试到福州正规的考场去考，后面学PHP以及其他的也照着这个思路去学。

张肖敏还说，其实网上有很多的视频，基本上自己认为比较好的PHP以及跟PHP相关的，包括Linux系统基本上都看过，现在互联网上能找到比较好的他都看过，通过这种方式去学，从没有到专业的培训机构去学过。

和张肖敏谈到自学困难的时候他说有很多，他认为这些困难可能是在自学考试前面就已经形成了一种习惯，这种习惯慢慢成为一种意识。书读百遍其义自现，张肖敏说，搞不清楚这个东西是什么，但是他就来回看。特别记得以前学数据结构、计算机网络，这个东西非常抽象，怎么看都不明白，先往后看，实在看不下去了就从最后往前看，看了一点也不行，又开始从前跳着看，换一章再看，用这种方式，来回倒，这有点像战场上攻城，先从一个角度出发，发现攻不下来，就换一个方向，继续攻打，再遇到挫折，接着换一个方向，总是这么不断尝试，后面总是能攻破这个城。因为困难是固定的，但是耐心是可以无限的，最后自然会有一种书读百遍其义自现的感受，完全明白这个东西。

在后面学PHP解决问题的时候，也会经常这样用这种方式碰到问题找不同的方案，比如一个网站方案的问题，有的写了但情况稍微有点不同，或者可能没考虑到一些因素，在后面一个问题就会找不同的答案，把它们凑合一起，用这种方式来回的去看，最后都能把问题解决。这也是张肖敏在个人在学习的过程中慢慢总结出来的一个自学的技巧。

同时张肖敏还认为有很多像他这样的人，在学习的时候，做某些事情会更好，其实有两个原因，一个原因就是他刚才说的，受外界干扰小，这已经创造了一个客观的因素。还有一个因素，拿他自己来说，是生活一直都有很大的压力感，这种压力感驱使他不断的向上，压力感越强向上的驱动性就更强，这两个方式主观跟客观结合起来，就会导致有时候他们做一些事情能比一般的人做得更好。

在找工作之余，张肖敏差不多每天能保持两个小时的学习时间，这个学习时间不仅仅是看书，比如他刚才说的视频教程，可以利用穿插的时间，可以边休息，边放着视频，有的时候看眼睛累了就听一会儿，基本上能保持两个小时的时间。包括他2012年9月专门去考了英语PETS4，相当于大学英语6级的水平，他认为在实际的使用中，接触英文技术文档比较多，后面专门强化了一下。包括他简历里面写很多的技术，其实是他之前项目里面完全用不到的，完全是属于他对个人的技术追求，学了很多东西，他认为对后面的成长很有帮助。在福州这么多年已经养成的一种习惯，恰恰是因为他认为有挑战才有未来。

自力更生

当记者问到肢体不方便能否影响到现在的一些编程工作时，张肖敏回答说，其实客观的来说编程跟这个没关系，但是有些小的方面他认为还是有一点关系的，这种关系体现在环境，也就是说如果有

一个无障碍的环境，但有的时候他们不具备这个，就像他之前在福州工作的时候，比如说门会有点卡，老板为了这个东西就要解决这些小问题，包括他自己在的那个区域可能要大一点，因为要方便轮椅来回转，不过这个是一次性的东西，不需要每天都要去解决这个问题，除此之外，他个人认为在实际工作中没有其他的影响。

而刚出来找工作时他对记者说以他的整体水平在福州找工作，基本上有一个概率的问题，有一些是主观上的原因，可能它本身不接受，有些是客观的，无障碍相对不太好，这些客观的东西不具备。他认为现在在福州找工作差不多是3%到5%的概率，一轮投下来大概福州有200份岗位，最后能够有5到10家是能够选的。

如今对于PHP开发张肖敏已经全职做了3年，记者问到在这3年里有没有后悔过选择这个行业时，张肖敏没有斩钉截铁的说没有，他觉得这条路越走越清晰，包括从一开始认为学PHP觉得他自己能做网站，后面清晰的认识到PHP这个方向能够做一些大型的互联网的项目，包括到后面跟进了整个Lamp体系，他自己在学习的过程中也在坚持整个Lamp体系，是一个很好的方向，互联网上很多应用和整个项目，这是一个整体的方案来解决。

对于他自己在工作中的一些成果，这就是张肖敏在福州时最后一个开发的B2B平台，还有他们开发的一个网址导航平台。在那边张肖敏是项目部的负责人，手下一般有两到三个刚毕业的或者有半年到一年经验的，任务就是由他下发，其他人只要解决模板层面的东西就可以，底层的代码都是由他自己来写。比如说用一个cms系统搭一个企业站，就直接分给其他人就可以做了。

在平常生活中，张肖敏和同事们非常融洽，平时在公司交流没什么障碍，该怎么聊怎么聊，该开玩笑开玩笑。小公司有的时候经常会有聚会活动、K歌，像这样的活动基本上都去，因为同事们有的时候要带我，需要挑一个能够轮椅上的去。在过去3年里基本上同事们都带上他。在平时，有一些女孩子对于网络不太熟，有一些其他的问题，会经常问他，只要是技术相关的问题都比较熟，所以跟同事们相处的很愉快。

离职原因

张肖敏跟记者谈了一些离职的原因，归结为一点就是发展受了限制，他在找工作的时候也是这种问题，比如说想找一个他认为平台稍微好一点，能够做一些比较深入的东西，这样的平台往往不接受他。所以他只能找一个愿意接受他的在那边做，做了以后就会发现，因为他自己的这种状态包括到现在一直都是，在当时也属于整个自学考试的过程中，家里人反对自学考试，在这个过程中承受非常大的压力，终于有一天能够找到一个工作，他能够抓住自己的命运的机会，在这种状态爆发的力量跟压力感、动力跟他们完全不一样。

在之前公司里面做事，可以说他整个的工作时间都是最长的，只要有事，包括个人的学习，很快他自己就发现业务逻辑跟不上他成长的速度，他还一直做模板层，其实已经把模板做腻了，张肖敏说他想写一些顶层的东西，一开始给人家建站，但是公司接不到大单，一直做3000、5000块钱的东西，做一段时间就觉得很没有意思，一直都没有接大单的机会。

所以他就想再去找一个比较好的，找的过程也是因为这个情况，想找个好的，人家也不要，只能找愿意要他的，刚好又是比这个强一点点的，然后我就换了。最后一个工作，那个时候再找■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201306/396716.htm

编者按

如果有跟我相似情况的同仁，建议还是多做分析参考，各种因素分析之后再做出自己的判断，毕竟鞋合不合脚，只有自己清楚。创业没那么简单，别一冲动给自己的将来埋下隐患，一定要冷静再冷静……

IT民工创业之殇

2005年从兰州大学计算机专业毕业之后，一直做ERP的开发和实施，凭着没日没夜的工作和还算灵光的脑瓜子，在开发和实施两方面都做的可圈可点，经手的项目有几万的，几十万的，也有好几百万的，幸运的是，做的这些项目都比较顺利，能按时实施完，能及时回款。职位也从小虾米到部门经理，一直都算顺利。

由于老婆一直在石家庄，而我的第一份工作在西安，因此工作5年，跳槽两次，从西安转战北京，在从北京到石家庄，迂回到达目的地。

以上这些经历，在其他日志中再说，本文只从2010年开始写起。

1、曾经小辉煌的业绩

2010年给华药做完医药营销项目之后，由于工作表现好被华药留下了，进入了华药销售分公司。这下好了，跟老婆团聚了，华药给的待遇不错，工作挺轻松。正好房子也装修好了。发现老婆也怀孕了，家里要添丁。多喜临门，各方面都挺满意，从此我们一家人开始了幸福的生活。

进入华药之后在总监的带领下，在信息化方面做了一些深入的管理方面的应用开发，现在看来是非常成功的。

首先是在医药营销项目的基础上，开发了费用预算管理子系统。把日常费用、高开费用和政策费用等凡是在公司产生的费用全部纳入系统管理，管理流程从预算开始，费用申请、权责数据的计算直到费用报销。每个部门发生的各种费用都进入了系统管理。

高开费用和政策费用的管理是最有效果的，仅这两项费用的计算就减少了办事处非常大的工作量。未上系统之前，办事处由专人在月底花几天的时间核算费用，时间紧，工作量巨大，核算之后递交给事业部核算员进行复核，最后在报销时递交给财务。使用系统之后，每月底由事业部核算员（或数据组）在系统中计算当月产生的各项费用（计算规则已内置，核算员只需点击计算按钮），审核之后即为办事处可办费用，办事处在报销时挑选已计算好费用额度的可报销发货单即可。这样一来，办事处就撤销了核算费用的专职人员，全国60多个办事处，在提升工作效率和精简人员方面的效果非常明显。听到不止一个办事处主任的赞誉之词，对它们来说太方便了。

在财务管理方面，不仅有原来收付口径的数据，而且每月底计算完高开和政策费用之后，即产生权责口径的财务数据，按照财务总监的话说，这对财务管理来说，管理提升幅度相当之大，他花了好几年时间试图建立权责口径，起初成效不大，现在用这个系统，终于完成心愿了。在后来的集团管理创新项目申报时，公司做了评估，该系统给公司创造的效益至少在四千万左右，项目产生了实实在在的

经济效益。

其次是在营销系统基础上，打造了集团的数据信息平台，曾经困扰集团和股份的数据收集问题，在这里得到了完美的解决。原来集团或股份需要收集各子公司数据时，没有有效的工具收集、合并、汇总数据，但是在这个平台上，定义好采集数据的维度和模板，各子公司只需导入自己的数据，系统会自动汇总形成他们想要的报表。看似很简单的功能，但在管理的提升方面是有目共睹的，现在集团和股份的人员几乎都知道销售公司有个赫赫有名的营销系统。

还有一些是小业务模块的应用，使用效果也不错，比如质检报告管理，远程订单管理等等，这些模块虽然功能不复杂，但是非常实用，在提高工作效率和简化工作难度方面效果显著，特别是汇总统计各种业务数据时效率非常高，给公司领导的决策可以快速提供准确的统计数据。

原计划完成费用预算系统之后，开发一套与商业公司沟通的商务平台，目的在于减少商务人员的事务性工作，把大量的无工作难度但很费精力的工作由系统处理。我们认为，在基药模式下，这个系统是一种压缩销售费用，提高管理效率的非常有效的管理工具，但是由于总监的调离，原计划就此搁浅，工作进入了另一种状态。

2、进入迷茫期

时间到了2012年的年中，儿子一岁多了，健健康康，活泼可爱。跟老婆俩一万多的月收入，在石家庄这样的城市，过的还是挺滋润的。考虑到家里离单位挺远，所以驾驶证一下来买了辆车子，虽然不是豪车，但是代代步、家用足够了。总之，生活上没有任何问题，很幸福的生活还在持续着。

在工作方面，总监调离之后，在销售公司再也没有懂信息化的高层领导了，再没有人能发动信息化项目的需求了，没有了“引擎”，我们只能转入系统的维护及更深入应用的小模块开发工作。

由于系统都开发的很完善了，各个部门的应用也没啥大问题，所以有那么几个月我相当清闲，每天几乎没有正经活。早上到公司后看几个小时的新闻，再偷偷打打游戏，中午跟老师傅打一小时的乒乓球，下午再找人侃侃大山，上网看看小说、笑话之类，一天就这么过去了。

这样清闲久了，逐渐产生了危机感，难道年纪青青的就这样过下去？同时也发现，没有了实际工作的磨练，原来的技能是明显下降的。更让人头疼的是，这几年通货膨胀的厉害，每月的工资用起来有点捉襟见肘了。跟同学的几次聊天，彻底让我清醒，再不能这么下去了，一定要再拼拼，要么升职，要么多多赚点儿钱。开始琢磨现在的处境是否正确。

在销售公司有升职的可能呢？答案是否定的。在我们信息部有部长，部长之上是总监，而部长是由集团任命的副处级主任工程师，根正苗红，有这种资历的人在销售公司不出三个，级别相当高。只要部长在销售公司一天，都不可能会有我的出头之日（仅仅说自己的前途，我对部长没有任何成见。其实他也是很憋屈，很高的级别，却只负责一个三级部门，管着4，5个小兵。在集团，跟他一样级别的，都是有这相当大的权利，都是各子公司的大员）。另一方面，我们信息部门是服务部门，非主营业务部门，升值通道本来就很窄，这条道走不通，意味着在销售公司我是没有任何升职可能了（除非从业务代表开始从头干，脱离信息化，做药品销售，但是谁会这么做呢？毕竟不是从零开始）。

按常理，在华药这样的大集团，下属几十个子公司，申请调到任何一个子公司负责信息化都不是难

题，但是我有我的苦衷。当初进入华药，总监给我的职位是专门创造出来的，虽然是普通员工，但是待遇却是部长级的，而且销售分公司的待遇是整个华药集团里最高的，也仅有销售分公司是按岗定资的（其他子公司都是按职称定资），那么去别的子公司，必然是薪资下降，这点我无法接受。

两条道都堵了，那么退一步，一直这么待下去，会怎么样呢？在华药这样的大国企，只要不犯大错误，不可能开除员工。同样，涨工资也不是说涨就涨的，十年八年涨一次就很不错了。分析来分析去，得到的结论是：只要不离开华药，那就拿着现在工资一直到退休吧。

这就是我的宿命吗？不到30，应该还能再拼拼吧，要不离职再闯闯，或找人自己立个山头！有了这样的想法，开始做家人的工作。这是个让他们很想不通的事，很稳定、几乎是铁饭碗、收入也不错的工作，为什么要离职？各种不理解，于是很长一段时间里，给家人‘摆事实讲道理’，‘旁征博引’分析现在的处境，还好以我这三寸不烂之舌，说服了他们，最终得到家人的支持。

没有了后顾之忧，开始找工作。简历放网上之后，石家庄好点的软件公司几乎都通知我去面试过，应聘了一圈，发现按石家庄的消费水平，他们能给我的待遇，最高的也只是跟华药差不多。相当郁闷的事。接着就是各种各样的纠结，看着手头的好几个Offer，到底是去还是不去呢？到底是离职还是不离职呢？

3、走上创业之路

还得提一提从销售公司调离的、我们曾经的总监。他离开销售公司之后，一直负责股份公司的内部市场化改革及系统实际应用，但是由于高层的政治斗争，项目做了一半，不了了之，好的建议得不到采纳，也是郁郁不得志。

直到有一天，有机会与老领导坐一起聊天，谈到了将来的打算，竟然不谋而合的想到合伙创业，既然好的管理思路不能在公司应用，何不自己开发软件，再应用到企业中呢？

既然要创业，得有发展方向，虽然很明确了是做管理类软件，但是做哪方面呢？这个过程是相当长且痛苦的，刚开始没有太在意自己最熟悉的行业，反而跑去琢磨教育类软件，养老院相关软件，诊所管理软件等。而且在每一个方向都琢磨了很长的时间，甚至都拿出了初步的方案。但是到实际操作时，发现没有相关行业的资深人员参与，做出来的东西根本没法用。无数次的碰壁之后，又回归到我们最熟悉的药品制造企业。

几个志同道合的人，经过无数次的讨论，最终确定了我们的发展方向：结合当时最急迫的全国药品招标，引入全程价值链的概念，计划开发一套在药品投标、标期全程起作用的利润优化及管控系统。

系统的管理思路闭环由老领导凭借其几十年的管理经验提出，再经过我们不同知识背景人员的补充形成。经过可行性分析、管理思路完善之后，从大的框架到实现细节，历时几个月终于完善了。然后拿着方案跟业内的管理专家和企业高管沟通，所幸的是在沟通的过程中所有的人都很赞同我们的东西，觉得对企业的帮助应该非常大。OK，有了这些东西，接下来是系统的开发了，这个工作肯定是，也只能是由我来完成了（几个合伙人中只有我是计算机专业出身，其他几位是管理、财务、销售方面的）。

与此同时，我们的公司也注册下来了，除了我们占股之外，也引入了华药元素，算是正式开始踏入了自负盈亏的创业之路。

4、系统初步规划

按照我们的规划，系统最起码应该有这几个功能：

1，能快速开发出各种漂亮的报表。这套系统主要是给企业高管使用的，数据的展示必须简洁美观。

2，审批流功能。部门级应用的模块中，要加入简单的审批流。

3，工作提醒功能。设定各种条件之后，系统自动提醒相关人员。

4，方便的数据导入功能。由于这套系统是一个跨业务平台，很多数据取之于不同的管理系统，有的来自ERP，有的来自财务软件，有的来自CRM等等，那么简单高效的数据接口是成功的关键。

5，严密、完善的权限安全控制体系模块。由于系统管理的数据是企业的核心机密，一旦泄漏给竞争对手，那将是毁灭性的，因此要从各个方面控制权限。我们规划至少应该从这几个方面控制：

1）服务器的硬件隔离。数据库及应用都在企业内部网，不考虑B/S架构，而且服务器的连接要加密，不能让管理员之外的人员知道。

2）系统登录使用动态口令。考虑给系统使用人员都配备动态口令牌，登录时除了需要输入静态密码外，还需输入动态口令牌中的动态口令。账户和口令牌一一对应，口令牌半分钟或一分钟变更一次口令。

3）系统登录与IP地址或MAC地址绑定。对于关键岗位人员，能登录本系统的IP地址或MAC地址必须是有限制的。

4）严密的授权体系。首先必须要有部门、角色、人员的三级授权体系。其次要能做到原子级授权，即各个功能的各项操作都要有权限控制。最后要有数据权限控制，不同的人打开相同的功能，其能看到的数据是不同的，有得多有的少。

6，快速开发。除了系统平台的功能外，大量的是业务实现功能，开发这些功能时要能快速实现，要是一个中等复杂的功能需要好几天的时间开发，那么等系统开发完，就好几年以后了。

按照这样的思路规划完系统后，我们突然发现，这对我们来说几乎是不可能实现的。像我这样的非专业开发人员（工作以后，很大一部分时间在做ERP实施，开发做的不是很多），能否将这些需求设计出来还是一问题，谈到开发出来简直是天方夜谈。自己开发不出来，是否可以考虑招人呢？也不行，手头的开发资金预算有限，没有办公场所，招了人去哪办公？又没有太多的精力管理开发人员？一系列的问题目前是没办法解决的。

唯一可行的还是硬着头皮自己开发，没招了。

5、开发平台选择

只能是购买一个现成的平台，在此基础上开发我们的系统。既然是购买平台，那就要慎之又慎，我们几个人都把身家性命、前途未来押在这上面了，不管哪个环节出问题，都将是致命的。出问题不仅仅是金钱的损失，原来的同事怎么看，丢不起人，三四十岁的人了，也输不起。处于这种考虑，详细讨论了好多次，最终形成了我们的需求。

1，平台必须要提供源码，不加密，开发的产品在销售上不能有任何约束，不用购买其任何授权。很鲜活的例子，有同事的公司曾经购买了别人的平台，开发时需要供应商授权■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201305/396581.htm

专访Tom：GenMyModel在线UML建模工具背后的敏锐团队

■ 设计模式的英文是Design Pattern，模式是Pattern的汉译。就是一种规则，或是一种模型，或是一种习惯。Pattern这个东西到处都是，并不只有技术圈子里才有。

【51CTO专稿】UML建模能够使程序员加深对整个系统的认识，并且通过模型与实际系统的对比能够及时发现系统分析，设计中的的错误。通常很多开发人员和架构师在使用更是为了便于团队之间的交流以及管理，也降低了开发成本以及风险。

需求带动着技术的发展，UML的建模工具也越来越多，但是大多数UML工具都基于Eclipse，例如大家熟悉的together、UML2、eclipseUML等等。可是开发者通常都要通过安装，部署才能使用。而GenMyModel致力于一个在线UML工具，在原本并不适合做UML开发的Web环境上成功推出了Web版UML工具。

因此，51CTO记者通过联系了GenMyModel的CTO Tom给网友们分享了新型在线UML建模工具的一些优势以及未来发展趋势的一些观点，同时也分享了他们背后敏锐的团队历时九个月研发首个beta版本的新型在线UML建模工具。



GenMyModel团队CTO Tom（法国）

能否简单的介绍下你们目前的团队？

Tom：我们的开发团队一共由八个人组成，成员包括：软件工程师，研究人员，社区经理和项目主管。可以说，GenMyModel的成功源于它背后敏锐的团队。

GenMyModel是一个专为软件架构师和开发师设计的在线建模工具。这是第一个携带输出和代码生成功能的云端UML工具。目前，GenMyModel还在测试阶段，在我们的主页中文版免费注册后，您可以即刻体验在线建模。从2013年开始，团队就收到了大量来自用户们就关于如何提高工具性能的反馈。而团队中也有一位专门的社区经理，负责每日和大量中国用户交流如何完善GenMyModel。

近日，两位在线建模专家与一位研究实时协作的专家合作，还有一位模型驱动架构方向的工程师专注于代码生成和“自动化构建”。 GenMyModel团队的CEO，计算机博Alexis Muller先生也会直接参与帮助技术团队的工作中。当然，还有擅长面向对象编程领域技术的团队！



能否聊聊对目前创业的一些想法？（为何选择在线UML这块创业）

Tom：总的来说，软件建模阶段成败直接决定了开发项目的成败。然而建模需要在最大程度上满足软件开发者的需求，如今还没有完全实现。 目前存在的，都还只是些针对专家的，安装使用的工具。

所以，我们认为完全有必要来开发一种可以“打破陈规“的工具，在全世界的开发者都可以轻松享受便捷的建模!

项目初期，我们把创新重点放在生成代码上。经过团队在实验室的研发，使该技术的拖动和拖放模块代码生产链的构建成为可能。我们想要把模型驱动架构带给所有的开发者,然而，市场调查表明，软件开发者对这一创新有些措手不及，因为他们更多在等待更先进的建模工具。

但是，决策者们却被高效生产率带来的收益所吸引，生成代码涉及建模和开发者强调安装式UML建模工具的主要问题。针对这一方向，我们跟进了研究并且调整了技术。经过软件开发者、构架师、项目经理 和决策者对这一创新的投票表决，我们的首个在线UML模型初成雏形。此后，庞大的用户群组成了推动GenMyModel前进的力量。每天，我们都能收到很多用户尤其是中国用户的反馈，有了他们的帮助，GenMyModel的使用便捷程度和各项功能也在不断完善。

从1997年OMG组织发布统一建模语言到现在，已经有近13年的历史。现在做在线UML工具会不会晚了?

Tom：现在，有越来越多的软件项目要开发，有越来越多的IT框架需要构建，对UML的需求也空前之大，所以。我们也在UML和软件建模的历史上跨出了崭新的重要的一步。设想一下仅通过简单的点击鼠标，模型就在构架师和开发者的手中成型的场景。

真正的建模工具、桌面工具他们大多数都依赖于框架或技术环境，不能在云中进行部署和扩展。现如今，谁能通过一个UML模型来实现真正的协作呢？要耗资多少才能实现分布式架构UML模型的分工协作呢？显然，唯一的途径就是使用在线模式，让UML工具满足开发者的需求。一些画图工具的出现已经为这一问题作出了回应，然而，对于软件开发者和构架师来说，他们需要真正的、适合的UML模型，并且用它们来生成文本和代码。

历时1000多个日夜的研发，GenMyModel团队在原本并不适合做UML开发的web环境上成功推出了Web版UML工具，并且还将继续创新。创新，在建模领域的专业知识以及围绕GenMyModel构建和日趋发展的社区是我们目前成功的重要因素。

你们团队开发的新型在线UML工具中的“新型”有什么含义？和其他在线建模工具有什么不同?

Tom：对于我们而言，所有UML工具都应该具备三个主要需求：简单，快捷，协作。没有什么能比基于网页的工具更好的了。但是，这也带来了庞大的技术难题。我们的想法完全区别于重新开发一个在线画图工具，因为已经有类似的不错的工具的存在。

例如，我们必须自己开发我们自由的模型存贮系统。那些例如CDO (Eclipse-based model repository) 这样的基于 Eclipse 和 EMF (Eclipse Modeling Framework) 的技术并不适用于网页模式。其原因是一部分代码（启用Javascript）必须在客户端的Web浏览器中运行，而CDO不用于这部

分代码。

经过很长一段时间的研发，我们的研究结果带来了GenMyModel的创新——改善建模方式的web版。这点我们每人经历过的例子就可以说明：无需花上整整一天在电脑上安装工具，5秒之内即可开启；无需花三天理解如何生成代码并推入Github，没有什么比这更直观的Web安装程序；一键即可和同事共享您的模型。

在线UML工具这个平台你们目前翻译了几种语言推广？你们看重哪个国家对UML技术的需求多些?

Tom：这种工具被世界上所有有建模和开发需求的国家所青睐。到2013年6月，GenMyModel已经被来自95个国家的用户所使用!其中欧洲、美国、中国、印度以及巴西占了前五位。

值得一提的是，我们有幸收到了很多来自中国用户的反馈（建议，bug需求和特点需求），这更说明了我们为什么要更加完善我们的网站。来自中国用户的建设性反馈之多，超乎了我们的想象，我们的社区经理和技术团队也在根据回馈不断完善我们的服务，让GenMyModel在更大程度上适应开发者的需求!

目前在线UML建模工具属于beta版本，以后有没有考虑加入更多的demo和支持更多的面向对象语言?

Tom：是的，我们想要更多的用户。用户为他们希望使用的代码生成器投票（在应用中的反馈区域），我们也在路线图中设置了一些优先权。例如，我们引用C++ et C#。当然，我们也会加入其它个人或者团队的代码生成器，例如，如果您有一个足够通用的从UML到PHP的代码生成器，我们将认真研究并很乐意把它加入到GenMyModel中。

有第三种可能性：在您的账户中加入您自己的生成器，您可以选择分享或者不分享。如果您有任何想法，欢迎随时和我们联系!

目前你们团队开发的在线UML建模是免费使用的，有没有考虑盈利?

Tom：GenMyModel面向公众，目前还在测阶段。我们想要给软件开发和构架师一二个最好的工具。推动我们前进的，是用户所给的反馈，所以至少2013年，GenMyModel还会保持测试间断。

我们也希望所有人都能有机会又快又好地建模。不过在这里要给出一个事实：随着以后功能的提升，我们将提供一些付费服务，但是即使是未来，我们的免费版本还是会一直存在。

目前UML技术人才还是比较缺失的，在开发UML过程中遇到的一些瓶颈（困难）如何去解决?

Tom：情况并非如此，无论欧洲以内还是以外，都有很多计算机工程师学校开设建模尤其是UML的课程。所有学生和工程师在学习编程前就了解到设计的重要性。 而UML是实现这一步骤的最好的标准之一。所以，我们见到了很多热情的软件开发者，他们有意愿在浏览器中编写并管理模型。 而云端建模为此提供了全新的协作可能！■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201306/399225.htm

新型在线UML建模工具（beta测试版）地址：
http://www.genmymodel.com/cn/
同时也希望各位网友在体验过程中积极反馈，致力打造更好的在线UML建模工具。

将死的平台 —— .Net

■ 近这些年我一直用C#开发。我认为C#和.net是非常棒的软件开发语言和平台。我感觉C#正是我想要的语言，用它即能开发上层东西，也能开发底层代码。然而，不幸的是，整个世界并不和我有相同的感觉，你可以感觉到.net的活力力正在慢慢的消褪。



最近这些年我一直用C#开发。我认为C#和.net是非常棒的软件开发语言和平台。我感觉C#正是我想要的语言，用它即能开发上层东西，也能开发底层代码。然而，不幸的是，整个世界并不和我有相同的感觉，你可以感觉到.net的活力力正在慢慢的消褪。

我有一些理由来说明为什么事情正在像这样发生(跟编程不相关)

1、固有的，人们讨厌微软，不管它做什么——即使是好事。这是很悲哀的事，我认为人们的态度应该立足于技术层面，不能总拿开源还是封闭说事。

2、Windows 8和WinRT对.net的支持让人困惑。即使.net程序员也是如此。给人的感觉像是微软创造了一个很酷的孩子，但却不愿让人们去喜欢它。

3、所有的windows手机，尤其是WP7(没法升级到win 8)，都很失败。从商业角度是失败，从对开发人员的支持上也是失败，对于一个平台，如果你不知道一年后它将何去何从，或你获取不到任何支持，你怎么会花大量的时间去研究它。

4、愚蠢的开源方式，我说的是，你可以阅读它的源代码，但不能出力，你无法发送pull请求。还有，微软什么事都要自己弄一套(Monorail vs MVC, Nunit vs MStest, NHibernate vs EF, 等等等等…)

5、平板电脑方面，微软没有取得多少市场占有率，全都是苹果和安卓。

6、对个人来说，它干掉了XNA。一个多好的SDK，突然就被抛弃了。为什么？没人能给出一个原因，你知道吗？

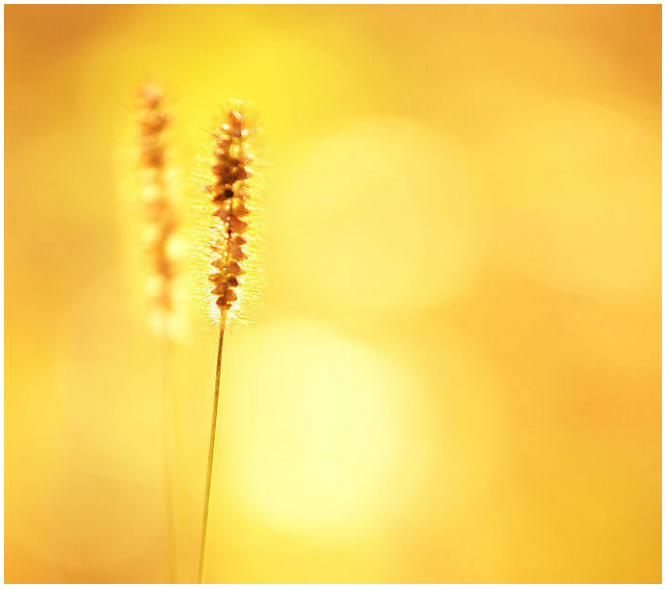
7、网上谣传的XBOX vnext也是只给XBOX的棺材上共享了一颗钉。

谁能挽救（可能的话）？

1、很诡异，也许可能是Xamarin，因为它的神奇的基于mono的移动开发工具。Unity也有帮助，但我疑惑他们究竟是怎么打算的。

2、外星人？

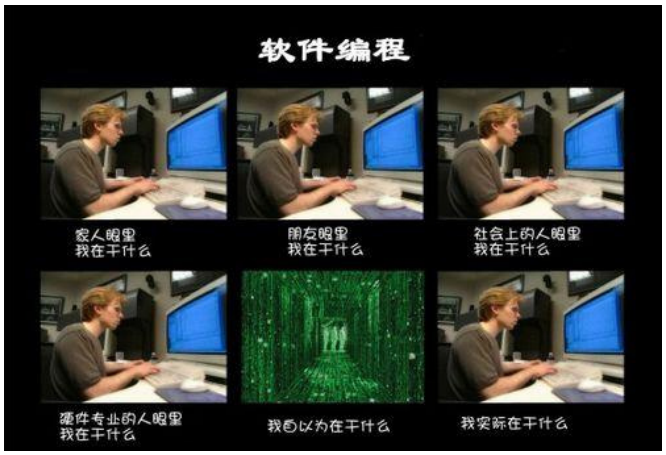
■



代码的世界可能是昏天暗地的，但是我们的思维不能这样随之混乱，否则一切都会前功尽弃。所以我现在编写程序的时候，经常…

编程思想：我现在是这样编程的

我在做什么



曾经，我试过接到一些需求。一眼带过后，脑袋马上随着高昂的斗志沉溺在代码的世界中，马不停蹄地敲着键盘直到最后测试的完成。我从思绪中恢复过来，乍一看自己写的功能，和需求差了十万八千里，我TM都在干嘛？

除此之外，我还见过类似的很好笑的事情。有一个程序员，经理提了需求，然后他在那里折腾了一天。结果不但没做出来，而且和实际需求都是完全搭不上调。经过询问发现，他不知道经理说了什么，也不知道自己到底在做什么。

代码的世界可能是昏天暗地的，但是我们的思维不能这样随之混乱，否则一切都会前功尽弃。所以我现在编写程序的时候，经常会想一下：我要做什么，我在做什么。更好的方法是把详细需求落实到文档，并时刻核对文档。

大局为重

2-8法则告诉我们，一个项目核心的功能只有很少，其它大部分都是对核心功能辅助或增强的。但当任务分发下来，我手头总有一些自己很想开发的模块，不过它们不属于那20%。我以前经常会在这些感兴趣的模块上花费很多时间和精力。

结果项目快要到上线期限，主要的功能却没开发完成，其它一些不起眼的功能却做得很好，但为此项目不得不延期了。如果反过来，只要对整体功能预期不会有太大偏差，可以将就的先上线。重要一点是：即使功能还有遗漏，但项目可以上线了，老板自然不会太追究，自己工作也能图个安心。如果不知道那些功能模块是最重要的，先问问经理。

人总是喜欢做一些自己感兴趣或者有挑战的事。不过在这方面，为了项目和团队着想，应该尽量压制这种诱惑。

性能永远不是优先考虑的问题

我从来不会一开始就考虑性能问题。如果项目成本很低，甚至到项目结束时，如果没有感觉到明显的性能问题，也不会去管。要知道现在已经不是

DOS的年代，CPU的计算能力很高，但成本很低了。重要一点是，如果只针对提升性能对代码做改动，很容易破坏代码的复用性和可维护性。而返过来，提高了代码的复用性和可维护性，则很容易提高性能。

下面有一个PHP的代码实例，功能是帮助用户重置密码（代码为了简单说明问题，请不要太在意一些无关的细节）

requestResetPassword是接收用户重置密码的请求并且做了相应的检查。为了更好的复用性，我将重置密码的操作单独分配到一个新的resetPassword的函数，更改完密码的后再调用sendEmail向用户发送一封通知邮件。

```
/**
 * 用户请求重置密码的接收器
 */
function requestResetPassword() {
    //检查用户是否存在
    if( !checkUserExists( $_GET['userid'] ) ) {
        exit('抱歉，用户不存在，请确认用户帐号。');
    }
    resetPassword( $_GET['userid'] );
    //最后向用户发送一封邮件
    sendEmail( $_GET['userid'], '重置密码成功', '
新的密码是xxxx' );
    exit('新密码已经发送到你的邮箱。');
}


/**
 * 帮助用户重置密码
 */
function resetPassword( $userid ) {
    //检查用户是否存在
    if( !checkUserExists( $userid ) ) {
        return false;
    }

    //进行重置用户密码的操作
    //略...
    return true;
}

/**
 * 向用户发送一封邮件
 */
function sendEmail( $userid, $title, $content )
{
    //检查用户是否存在
    if( !checkUserExists( $userid ) ) {
        return false;
    }

    //发送邮件操作
    //略...
    return true;
}


/**
 * 检查某个用户是否存在
 */
function checkUserExists( $userid ) {
    $user = getUserInfo( $userid );
    return !empty( $user );
}



/**
 * 获取某个用户的数据
 */
function getUserInfo( $userid ) {
    //假设我有一个query的函数，它用来查询数
据库并返回数据
    $user = query( "SELECT * FROM `user`
WHERE `uid`=" . intval( $userid ) );
    return is_array( $user ) ? $user :
array() ;
}
```

现在问题是，这三个函数都同时使用checkUserExists这个函数来检查用户不存在，数据库查询了三次，这样带来了一些额外的开销。

如果要去掉三者之间任意一个checkUserExists，

看上去是可能的。但是如果之后有某些功能要调用resetPassword或者sendEmail，用户不存在时，系统可能会发生错误。

还有一个解决方法是，将resetPassword的逻辑写到requestResetPassword里，再过一点，把sendEmail的逻辑也写进去。这样函数调用减少，数据库查询也变成一次了，性能得到了提高。但是重置密码和发送邮件的功能将不能得到复用，并且违背了单一责任的原则，代码复杂度也提高了。

不过，因为函数分离和复用性都很好，如果实际性能受到影响，可能考虑用缓存的方法减少数据库查询，我改动了它们共用的checkUserExists函数：

```
/**
   * 检查某个用户是否存在
   */
  function checkUserExists( $userid ) {
     //增加一个缓存，用以记录检查用户的结果
     static $cache = array();

     //检查当前用户是否已经检查过一次
     if( isset( $cache[ $userid ] ) ) {
        return $cache[ $userid ];
     }

     $user = getUserInfo( $userid );
     //把结果记录到缓存中
     $cache[ $userid ] = !empty( $user );

     return $cache[ $userid ];
  }
```

也可以用同样的方法改动getUserInfo函数。

这里可以看到，当代码的复用性提高时，想提高性能是很简单的，性能的瓶颈也很容易被发现和修改。

尽管这个例子对性能影响还不够大，还有一些影响更大的，比如说遍历，我可能为了复用而将遍历封装到一个函数中，并且多次使用它。这些开销对我的项目根本没有预想中那样有太大的影响，或者说是微乎其微的。所以我更愿意把时间花在如何提高代码的复用性和维护性方面，而不是纠结于浪费多这一点性能。实际性能如果真的达不到要求，也可以权衡增加硬件配置。

名字长一点好

函数名和变量名等除了给机器看，也要给人看的，有时一个简单直接的好名字实在是很难想，这时不妨用长一点的名字更好。可读性更好：

```
//好名字
  class ErasedTypeEquivalence {
  }
  //坏名字
  class ErdTypeEqe {
  }
  //好名字
  function checkUserExists () {
  }
  //坏名字
  function ckUserExt() {
  }
  //好名字
  $result;
  //坏名字
  $ret;
```

■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201306/398758.htm

从100PV到1亿级PV网站架构演变

编者按

一个网站就像一个人，存在一个从小到大的过程。养一个网站和养一个人一样，不同时期需要不同的方法，不同的方法下有共同的原则。本文结合我自已14年网站人的经历……

一个网站就像一个人，存在一个从小到大的过程。养一个网站和养一个人一样，不同时期需要不同的方法，不同的方法下有共同的原则。本文结合我自已14年网站人的经历记录一些架构演变中的体会。

1：积累是必不可少的

架构师不是一天练成的。

1999年，我作了一个个人主页，在学校内的虚拟空间，参加了一次主页大赛，几个DREAMWEAVER的页面，几个TABLE作布局，一个DB连接，几行PHP的代码嵌入在HTML中，再用FTP传到服务器上就可以给别人展示一个网站。

2000年，个人主页已经不能满足好奇，在当时的网管中心管起几台机器，作起网线水晶头，用ALL PEOPLE SEEMS TO NEED DATA PROCESS的理论开始认识了7层网络模块（面试技术员工时，经常会问这些网络基础知识的理解）。有了基础理论的武装，我也开始配置各种服务来玩 LINUX，AIX和FREEBSD这些系统。面对各种原理不懂的系统，目的只是想尽办法去解决网站需要的各种基础服务。当时搭建了REALSERVER 流媒体服务，各种开源FTP下载服务，BATTLENET游戏网关，APACHE（keepalive等配置，http报头相关的知识 也是面试的老客户），DNS，QMAIL等服务给学校的学生使用；

网站有近10万PV的时候，开始考虑如何作扩展拆分，MYSQL的MASTER SLAVE作读写分离，MYSQL的索引优化是当时唯一会使用的DB性能优化方法。这个阶段基本上能解决需求，同时也遇到瓶颈，不知道访问量再大一个数量级，怎么办?明显感到技术能力不够。当时受限于网站的量一直没有新的数量级的突破，导致了几年内技术工作以维护为主，体验着网站日常运维的各种纠结，体力活。而这时期网站的2层架构方法也一再被重复应用到后续的N个网站的搭建过程中，2001年开始的JSP与PHP之争好像对我没什么感觉，因为我还是只会用那种很土的2层架构作着网站：页面中嵌入JSP，连接后端的MYSQL进行数据的CRUD，没有事务，没有考虑数据库读写错误，没有考虑两层系统不可用，因为有问题只需要重启，有报错只需要改JSP后刷新页面。作网站确实是体力活。

2005年，在道富参与了FXCNG项目，用MULE ESB，接触到MULE的前几个月还没有意识到这个系统在架构中的位置。直到半年后，在与第三方系统集成的应用中，才发现，这样的一个系统就是传说中的中间件，他可以专业的解决一部份系统的职能，比如当时的消息和远程调用代理。这时候我逐步有了一些架构观点，一个大型的应用系统中，是需要隔离一部份技术职能，让系统责任单一，方便技术维护和团队扩展，专人办专事。

2006年，阿里软件，一个全新的开始。进入阿里软件的前三个月在马总的老家，淘宝的发源地：湖畔花园小区，2楼的4室两厅里进行了封闭式开发，

很累，但很兴奋。这段时间，我参与了一个全新外贸ERP系统的搭建。虽然只作为一名普通的JAVA开发工程师，仅负责一个很小的模块开发，但还是正式体会到一个大型系统的初创过程。这段时间对MVC的架构理论有了深刻的理解。MVC三层架构比2层架构带来的更好的技术扩展与团队扩展能力让我映像深刻。M层负责DB 逻辑的CRUD，架构师们开发出了大量的中间层JAR包，完成了DAO层，DO的封装，M层也完成了SERVICE层，完成了BO的封装，接口包的封装，系统使用了最常用的工厂、单例、FACACE等模式（直到现在，我面试人还是常常只会问这些基本的模式）。V层的模板隔离，前端开发可以在这一层和后端开发一起修改界面（之后，更大规模的团队连这一层都可以再分离）。C层完成输入输出的基本校验和检查然后调用后面的服务层功能或作转发。 简洁朴素的结构生命力是比较持久的。直到现在，MVC还是具有很强的生命力，在更种大型网站中承担重要架构骨架。

07年，我对应用层，服务中心层，持久层三层架构并没有多少的实践应用和理解。因为MVC对于初创系统或中小型网站，20人左右的团队规模来讲，已经适用。换个角度看，当时在3个月内作出一个完整的ERP系统也只用使用MVC，再选进的架构也需要考虑网站发展的阶段。用户一边使用，工程师一边迭代改进架构，这个方式是作网站，不是传统应用软件开发。ERP本身源自传统应用软件领域，我们在用互联网的方式作管理软件，最大的挑战应该是这种边作边改进架构的理念。07年，这个理念之争逐步在团队内达成了一致，架构师们小心的平衡着业务和架构，这个网站高峰时，也支撑到了日访问量近千万的级别，没有闪架。

08年，阿里软件开放平台，又是一个新系统。全新的系统架构总是可以得到足够的时间来考虑未来1到3年的增涨。实际上，互联网系统，我们感觉只能考虑到一年的架构。这一年，我参与架构设计，开始理解了三层架构的价值与扩展能理，服务中心开始搭建，因为这个系统将接受的几百万在线的旺旺用户访问，所以部份系统开始考虑服务中心，想把业务逻辑聚合到服务层，由统一的团队进行扩展维护。另外，随着两年下来小的业务系统越来越多，小机上的oracle连接数也吃紧了，服务中心的需求越来越大。首先进行的是用户中心，想法是集成整个集团队账号体系，基于『旺号』体系。这个用户中心项目有个响亮的名字：UDB，项目过程可以写一本书。这里不再展开。

《SAAS架构设计》这本书，针对的是软件平台的早期ISV用户。现在的眼光来看，这书写的过于简单，正如十年后再来看今天我的写这个笔记一样。

09年，加盟了刚刚成立的aliexpress部门。经历了两三个应用的小系统到亿级PV的在线交易系统。遇到了很多问题，体会到一个小系统与大网站不同阶段架构的演变过程有不同的难处。

下文从不同维度分享亿级PV网站架构下我的体会和观点。

2：知识结构

网站架构师有很多，有科班出身的，有美术专业的，有生物专业的，有学物理的，有派出所警察出身的，我觉得都是OK的。我也接触到了这些架构师，非常有特点，在很多技术领域有自己专深的。英雄不问出路，好汉不提当年勇。架构师知识背景可以不同，个人看法是不同领域的人作网站架构可以带入很多交叉的思路。就像种树的人再去种花，其实也是可以看到一些共性的总结抽象。

网站架构师需要有编程的经验，从基本的算法，常用的设计模式，多线程开发，远程调用，不同类

型数据源使用，这些是面试的时候看得基本功。我认为一个资深的测试专家一定是开发高手，一个架构师必须也是有长期的开发经验，很多性能优化是要从一行行代码优化起的。试想在一个被调用1000万次次每天的页面，一行代码如果每次都走到，每次少运算1ns,也可以节省不少的电力。我为环保作贡献，我骄傲。

网站架构师需要对网络环境有很好的知识理解。架构问题是需要考虑网络部署。比如系统因不可用而发生切换时，从一个机房切到另一个机房，要考虑网站的服务对用户访问速度上会有多大影响。这时候的技术方案可能是切DNS，也可能是切前端的跳转机，或是底层部份服务调用到另一个机房。对于这类切换的方案，架构师需要计算网络时间的开销带来的QPS影响，和用户体验上的延迟，每个请求估算需要精确到ms级。如果是全球范围内DNS切换，需要知道DNS刷新的时间经验周期，比如：全球更新在1小时左右，而80%的地区用户会在20分钟内刷新，这样系统带来的业务影响会有多大。

网站架构师需要对网络协议有深入的理解。HTTP协议是最基础的，无论是SESSION还是COOKIE在HTTP协议基础上怎么应用，COOKIE的大小，数量，浏览器是怎么处理HTTP协议的。这些基础有关键时候会影响业务的进行。比如，SAFRI浏览器对第三方COOKIE是禁用的，某功能跨域写 COOKIE的时候每次都会重新生成COOKIE，直接导致系统统计用户UV的时候，数量增大，影响各种转化率的计算。HTTP协议还需要考虑本身的连接管理池大小和连接是否KEEPALIVE，这些细节很多时候成为架构上扩展能力的瓶颈。一个静态页面服务的HTTP MAXCLIENT设置 为2500，机器只有10台，很可能在一次中小型活动中连接数到顶，用户部份请求无法满足。

架构师需要考虑数据格式带来的性能影响。很多远程系统调用走的是HTTP协议为基础，数据格式为纯文本或JSON，或XML等，这类调用需要考虑数据的序列化和反序列化，这个工作是CPU开销型的，对性能优化上需要有针对性。QPS高的系统RT一定会短，但RT短的系统不一定比RT高的系统能表现更好的 QPS。

架构师需要有很好的数学能力，计算一个QPS里系统从网络请求发出，到网络的IO时间，DB的磁盘读写时间，CPU运算时间，再到数据库连接数，数据分库分表容量规划，都需要有精确的计算。因为容量计算不正确带来问题也是非常多的。比如一台小机上ORACLE的连接数开了2000个，而应用系统由于不断的扩展，小业务系统不断加入，大型促销活动前，临时机器的不断上线，很快就把DB连接数用完，引起业务部份不可用。架构师需要去合理估算每种应用的服务能力，以及他对DB等资源的合理连接数。

加构师对JVM的内存分区及管理策略要有深入的了解，GC的频率可以发现很多系统容量的问题。一个OLD区不断加大的系统，伴随YGC高频发生，加上 TCP机器连接数很可能高，可能是要是机器了。一个业务功能不断叠加的系统，很可能PERM区会需要加大设置，否则容易OUT OF MEMORY。

加构师需要精读《数据库系统概念》这类书，对不同DB的索引原理和库表存储结构有了解，我们可以不是ORACLE ACE，但一定要听得懂ACE的DB架构和性能优化方面的建议。并且在原则上，前端用户系统架构上不要出现直连DB的设计，这是亿级■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201305/396157.htm

编者按

今年1月初，20岁的洛翰·沙阿(Rohan Shah)收到了一封来自谷歌的电子邮件。谷歌，世界上最好的雇主，希望就一个实习生岗位对他进行面试。……

在Google做实习生是什么样的

今年1月初，20岁的洛翰·沙阿(Rohan Shah)收到了一封来自谷歌的电子邮件。谷歌，世界上最好的雇主，希望就一个实习生岗位对他进行面试。

沙阿就读于伊利诺伊大学。在学校的一次招聘会后，沙阿在线填写了谷歌的实习申请。几周后，他收到了谷歌的通知。不过，一个月之前，他已经获得了高通的夏季实习生职位。

然而，这是谷歌！沙阿不愿意放弃这一机会。



谷歌一名发言人曾表示，该公司每年在美国会收到4万个实习申请，但仅招收1500名实习生。这一职位的竞争非常激烈，而欧文·威尔森(Owen Wilson)和文斯·沃恩(Vince Waughn)甚至专门为此拍了一部影片。

沙阿走完了长达1个多月的整个面试流程。到1月底，他获得了这一职位。沙阿并未取消在高通的夏季实习计划。他打电话给学校申请暂时休学，随后前往了加州山景城，成为了谷歌的50名春季实习生之一。

那么，谷歌的实习生面试流程是怎样的？成为谷歌实习生之后又会怎样？沙阿和谷歌的其他实习生讲述了他们的故事。

接受面试：缓慢而冗长的流程

谷歌的面试流程，无论是申请实习生还是全职职位，都需要首先在网上填申请。求职者需要填写在校的平均绩点、过往履历，以及课外活动信息。沙阿没有费太大功夫就让自己的简历引起了关注。他会说3种语言，曾获得志愿者荣誉奖章，同时也是学校的校长奖学金获得者。此外，他还曾是一名助教，在其他公司实习过，并曾因兴趣开发过Android应用。

在沙阿去年11月提交申请的几星期之后，谷歌才做出回复。谷歌向他发出了电子邮件。

另一名实习生的应聘者埃文·卡尔米(Evan Carmi)表示，当他2010年进行申请时，他等了1个半月时间才收到谷歌HR的回复。在邮件交流之后，谷歌会与应聘者进行两次电话面试，而面试官是谷歌员工。

从这时，应聘流程才正式启动。

45钟的一系列技术面试

外界曾经流传，谷歌的面试官会在面试过程中问一些很难的问题。在遭到这样的负面报道之后，谷歌已禁止员工问一些古怪的问题，例如“加拿大有多少头奶牛”。不过谷歌的面试仍然很注重技术，即使对实习生也是如此。

谷歌对公司内部信息有着非常严格的控制，但谷歌允许我们对沙阿进行采访。沙阿表示：“你需要将知识应用在非常实际的场合中。”谷歌的面试会对一些问题进行深入讨论。他表示：“面试需要看出你是否有能力规划一个系统，或者你能否让一些事更有效率。”而最优秀的应聘者能在45分钟时间内迅速证明自己。

卡尔米接受的面试同样技术性很强，他曾在2010年的一篇博客文章中谈到了这些。他接到的第一个面试电话来自谷歌的现场可靠性工程师。这名工程师问了一些关于Python的问题，例如“写一个函数提供以下功能：输入一个列表，输出列表拷贝，同时移除其中的重复元素”。

卡尔米接受的第二次电话面试来自谷歌网管工具团队的一名成员，他是卡耐基梅隆大学的研究生。卡尔米被要求实际编写代码。在45分钟时间内，他主要就在解决一个问题。整个过程令他紧张得流汗。卡尔米随后还被要求接受第三次技术面，而他在谷歌的应聘之路也止步于此。

沙阿则较为幸运。在接受了两次电话面试之后，他收到了谷歌HR的邮件：“你在面试过程中表现优秀，我们希望继续。”随后，谷歌的一名招聘人员帮助他确定在哪一部门担任实习生，随后他将接受进一步面试。“我接受了5个不同团队的面试。”他表示。

与电话面试不同，团队内部面试与技术无关。这类面试帮助可能的实习生了解谷歌内部的不同部门，确定他们是否愿意在某一部门工作。

到11月底，即提交在线申请的3个月之后，沙阿正式成为了谷歌的实习生。他将加入谷歌的Android部门。他的下一站是：山景城。

住宿、室友和交通，谷歌全包

接到横跨美国的实习通知，并在两周后就开始工作，这似乎完全不可能。但如果你是谷歌的实习生，那么谷歌将帮你完全解决住宿和差旅等问题。沙阿入住了谷歌已付费的一处公寓，位于圣何塞的North Park。他的室友是谷歌的其他实习生。

沙阿有3名室友，两名来自阿根廷，另一名来自乌克兰。“我认识了一些来自完全不同文化的人们。我向他们学习，同时也了解了一些他们的语言。”沙阿表示，“对我而言，这是一次很棒的住宿体验。”他表示，这处公寓非常好，火车站就在附近，他可以很方便地前往硅谷大部分地方。

如果你成为了谷歌的实习生，那么汽车和自行车都不再必需。在整个加州湾区，谷歌提供免费的穿梭巴士服务，可以达到旧金山、圣何塞、Palo Alto和伯克利，帮助员工上班和回家。在谷歌园区内部，自行车可以长时间租赁，员工也可以骑车从一栋大楼前往另一栋大楼。

一周半的适应时间

实习生有一周半时间适应在谷歌的工作。实习生将了解数据中心如何运转，公司的职能部门，以及谷歌的目标是什么。他们也将见到其他新员工。“在第一周，你就会觉得自己是一名已工作一年的员工。”沙阿这样说，“你将很快适应公司。”■

本文未完，请阅读原文：
http://developer.51cto.com/art/201306/397444.htm

从面向对象的设计模式看软件设计

■ 设计模式的英文是Design Pattern，模式是Pattern的汉译。就是一种规则，或是一种模型，或是一种习惯。Pattern这个东西到处都是，并不只有技术圈子里才有。

什么是模式

在正式说明GoF的那23个经典的设计模式其实和OO关系不大并和Unix的设计思想很相似的这个观点之前，让我先来说说什么是模式？设计模式的英文是Design Pattern，模式是Pattern的汉译。所谓Pattern就是一种规则，或是一种模型，或是一种习惯。Pattern这个东西到处都是，并不只有技术　子里才有。比如：

文章有文章的Pattern。如新闻有新闻的Pattern（第一段简述了整个新闻），诗歌总是抒情，论文总是死板，讲稿总是高谈，漫画总是幽默的，……

小说有小说的Pattern。比如，

武侠小说必然要整个武林大会，整几个NB的武功和大师，分个正派和反派，还有一个或数个惊天阴谋，坏人总是要在一开始占尽优势，好人总是要力挽狂澜……

言情小说总是要有第三者，总是要有负心人，里面的女子总是要哭得死去活来，但又痴心不改，……

新闻联播的模式是：头10分钟领导很忙，中间10分钟人民很幸福，后10分钟国外很乱。

中国政府官方宣传稿也模式也很明显，各种赞美，口号，胜利，总是要坚持个什么，团结个什么，迈向个什么，某某精神，某某思想，群众情绪稳定，不明真相，等等……

春节的模式是，回家，吃饺子，放个鞭炮，给压岁钱，同学聚会…… 同学聚会的模式基本上都是在饭桌上回忆一下校园时光，比较一下各自的当前处境，调戏一下女同学…… ……

这就是Pattern，只要你细心观察，你会发现这世间有很多很多的Pattern。

GoF的23个设计模式

《设计模式》这本书中，GoF这四个人总结了23个经典的面向对象的设计模式，某中有5个创建模式，7个结构模式，11个行为模式。很多人都会觉得这是面向对象的设计模式，很多人也觉得非面向对象不能用这些模式。我觉得这是一种教条主义。就像《那些流行的编程方法》中的“设计模式驱动型编程”一样，就像《如此理解面向对象》一样的那么的滑稽。

好了，回到我的论点——“GoF的这23个设计模式和OO关系不大，并且和Unix的设计思想基本一致，只不过GoF用OO实现了它们”，就像我上面说过的那些生活中的Pattern一样，只要你仔细思考，你会发现这23个设计模式在我们的生活和社会中也能有他们的身影。而且也一样可以用OO的方式实现之。

让我们来看看这23个经典的设计模式中的几个

常用的模式：

Factory 模式，这个模式可能是是个人都知道的模式。这个模式在现实社会中就像各种工厂一样，工厂跨界的不多，基本上都是在生产同一类的产品，有的生产汽车，有的生产电视，有的生产衣服，有的生产卫生纸……基本上来说，一个生产线上只有做同一类的东西。这和 Factory模式很相似。编程中，像内存池，线程池，连接池等池化技术都是这个模式，当然，Factory给你的一个对象，而不单单只是资源，factory创建出来的对象都有同样的接口可以被多态调用。这其实和Unix把所有的硬件都factory成文件一样，并提供了read/write等文件操作来让你操作任意设备的I/O。

Abstract Factory，抽象工厂这个模式是创建一组有同一主题的不同的类。这个模式在现实社会当中也有很多例子，比如：

移动公司的合约机计划，88套餐（通话100分钟，短信100条，彩信，20条，上网200M），128套餐（通话200分钟，短信150条，彩信50条，上网500M）……

家里的装修，总是要有厨卫，有门，有灯，有沙发，有茶几，有床，有衣柜，有电视，有冰箱，有洗衣机……，这些都是必需的，只是每个家庭里的具体装修不一样。

Diablo游戏中的Normal，Hard，Nightmare，Hell模式，这些模式的怪和场景和故事情况都差不多，就是每个场景的怪物和装备的属性不一样。或是WarCraft中的地图就是一个Abstract Factory模式(注：Warcraft的地图什么都能干)。这和学校中的小学，初中，高中，大学差不多，都是一样的学习环境，一样的教学方式，一样的教室，都要期中考和期末考，都有班长和科代表，就是学的东西的难度不一样，但基本上都是语文，英语，数，理，化，还有永远都有的政治课。学校就是一个抽象工厂。

这就是抽象工厂的业务模型（或是：Business Pattern），你觉得是不是不一定非要用OO来实现这样的模式？（我们思考一下，我们会不会被先入为主了，觉得不会OO都不知道怎么实现了），不用OO，用相同格式但内容不同的配置文件是不是也能实现？在Unix下，抽象工厂这个模式在Unix下就像是/etc/rcX.d下的那些东西，1代表命令行单用户，2，代表命令行多用户，3代表命令行多用户完整模式启动，5代表图形界面启动，0代表关机，6代表重启，你要切换的话，init <X>就行了。

Prototype模式，原型模式，复制一个类的实现。这个模式在现实中的例子也有很多：传真，复印，都是这个模式。Unix进程和Github项目的Fork就是一种。进程fork明显不是OO的模型（参看：关于Fork的一道面试题）。用非OO的方法同样可以实现这个模式。

Singleton模式，单例模式。生活中，公司只有一个CEO，法律限制你只能有一个老婆，你只能有一个身份证号，一个TCP端口只能被一个进程使用，等等。软件开发方面，并不一定只有OO才能做到，你可以用一个全局变量，一个中心服务器，甚至可以使用行政手段来约束开发中不会出现多个实例。Unix下实现单例进程的一个最常用的实践是在进程启动的时候用“(S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)”模式打开一个“锁文件”。

Adapter模式，适配器模式。可以兼容欧洲美国中国的插头或插座，万能读卡器 ……■

本文未完，请阅读原文：
http://developer.51cto.com/art/201306/397482.htm

大话科技：产品经理/码农/设计师的3P战争

□ 快刀青衣/文

前言：这是一个真实发生的故事。在某个产品策划会上，一个PM面对来自内容部门和营销部门同事的会议上，骄傲的说：“根据鳄鱼法则，我们需要做的就是做一款服务于网友的优质产品，来扰动这个封闭的行业，增加大家的活力和竞争力。”

坐 在场下事不关己的我放下手机，自卑于自己的孤陋寡闻。悄悄打开手机，搜一下才终于明白“鳄鱼法则”指的是如果你一只脚被鳄鱼咬住了，那么不要去挣扎，那样 的话会让你连手都没有了。最好的办法是放弃你的脚丫子。而这个产品经理想说的明显是“鲶鱼效应”，这个词的意思就是把灰太狼扔到羊村里，羊村里的羊都会积 极上进热爱生活起来。场下很多小同学一脸懵懂一脸崇拜，我想他们会牢牢的记住鳄鱼法则，以备以后复述和装逼使用。

本着将“精英话题屌丝化低俗化”的准则，我将写一个“大话”系列，将精英话题们拉下云端。行文之处，过于随意，请专业人士不吝指教，精英人士绕道而行。

最近在无数场合，每当我和HR MM坐在一起，都不由自主的举起酒杯，说：“先干为敬！”她就一脸茫然一脸惊恐的看着我，我想她的潜台词一定是“我已经有孩子了，你也有孩子了！”

其实我只是有一个朴素的愿望，就是能在公司的劳动合同里加上一条“保证在需求发生无数次变化的情况下，设计师和程序员保证产品经理基本的人权，不进行残暴的肉体摧残。”

很多人不理解为什么产品经理/设计师/程序员这本是一个团队中的人，为什么总表现的犹如非洲动物世界里的天敌一般？

那么，在我的心中，这三者其实是一场发生在某旅馆房间大床上的3P战争。

一.什么是3p

科普帝来了。那些以为3P就是指3 people的同学请注意了，从严格意义上来说，3P指的是两男一女，并且特指一对有情感关系的男女与另外一个男子进行的共同的性行为。

文雅一点儿的称呼是“两马同槽”，所能找到比较早比较知名的出处是《红楼梦》第六十五回中，“贾珍、贾琏的两匹马同槽而不能相容，以至互相服踢”，两个男人争小姨子尤二姐，最后三个人开心的在一起了。

用图形表示3P就是“卜orz么”。还有一个很重要的衡量条件就是如果在过程中，女性只是起到性玩物的作用，那就不是真正的3P，而是聚众淫乱。

言归正传，诸位不觉得产品研发的奥妙正好似产品经理/程序员两男和设计师一女共同达到人生大和谐的过程吗？

再科普一下三个名词解释，设计师：臭画图的！程序员：臭码农！产品经理：指手画脚瞎指挥的臭SB！

二.三者之间的关系

其实不管产品经理/设计/程序员三者怎么斗，都是属于一个团队的家务事。在遇到大是大非的问题时基本都能做到枪口一致对外，只不过大部分团队很难遇到大是大非的问题。

很多时候，最喜欢黑程序员的是程序员自己，最喜欢黑产品经理的是程序员和产品经理自己，最喜欢黑设计师的是产品经理和设计师自己。这是多么奇怪的三种人啊，都喜欢黑自己，但是又不满别人黑自己。

我决定构建一个模型来描述一下三者之间微妙的关系，而这个模型还真的被我找到了。

如果放在古代，产品经理就是皇上，设计师是皇后，而程序员就是开疆辟土的肱骨之臣掌兵大将。很多时候国家的大方向需要皇上去把关去指点去沟通协调，皇后需要 给朝廷撑起颜面，要不怎么会有母仪天下一词，而大将则需要去攻城拔寨，在战略上听皇上的话，在战术上将在外君命有所不受。

那么让我们看一下原本应该是和谐关系的三个人都会发生什么样的事情。

三.不同类型的团队案例

1.三方和谐型

盛唐之所以当年能万国来朝，与唐太宗李世民密不可分。但是只要有一个贤主就能抵达巅峰吗？一个重臣和一个贤内助密不可分，唐太宗外有魏征，内有长孙皇后。

这 个团队，魏征是个奇葩，在隋末，魏征先是投靠了瓦岗军李密，负责掌管文书，提出的战略不被重视，瓦岗军被王世充杀得大败。魏征就跟随李密投靠了李渊。然后 魏征为了在李渊面前出头，而不甘于当李密的小弟，主动请缨去收服李密以前的部下。后来去招降窦建德的魏征反倒成了窦建德的起居舍人。没过几天好日子，窦建 德就被李渊摆平了，魏征又被抓了。这时，他的名气吸引了太子李建成，他就开始为李建成出谋划策。过了玄武门事变，魏征的新主子又被李世民杀了，他又投靠了 李世民。这么坎坷的克主名气，没有能阻扰李世民的用人决心，也使魏征成就了一代名臣，而不是像吕布那样的三姓家奴。

而魏征曾直言不讳的说：“我希望你把我当做一个良臣，而不是一个忠臣。”两者的区别在于，良臣可以成就一番事业，使国富民强安居乐业，君主也有好名声，可以流芳百世。而忠臣，只有在乱世才能显出忠诚，要么被奸臣杀，要么被乱臣杀，要么被昏君杀，对君主的名声都不怎么好。

之所以啰嗦这么多，核心思想是：将程序员看做良臣，不要去苛求他去当忠臣。程序员和产品经理是不是朋友都无所谓，两者的共同目标是做出优秀的产品来。而不是去营造貌似和谐的氛围、人云亦云的探讨会。魏征的直谏成就一代盛世，这是良臣能做到忠臣做不到的事情。

长 孙皇后更是唐太宗的贤内助，最出名的事迹是朝服谏君事件。某日，魏征在朝堂上直言犯上，李世民气冲冲的甩下一句“誓杀这个乡巴佬！”但是回到后宫，说起这 个事情后，长孙皇后退下，一会儿换了最华丽最郑重的朝服恭贺李世民。李世民很诧异，长孙皇后说：“主明则臣直。”竭力为魏征开脱。

看，产品私下里大骂技术太强势不听话■

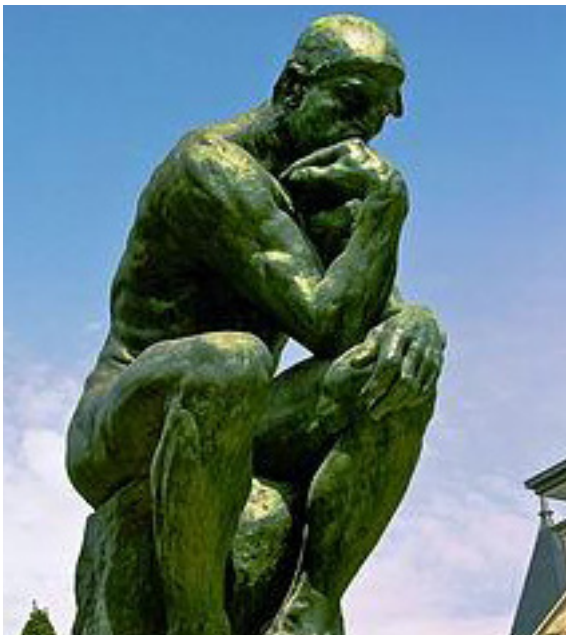
本文未完，请阅读原文：
http://developer.51cto.com/art/201306/397837.htm

如何留住你最棒的程序员

□ Erik Dietrich /文

变得沉着冷静

正因为我刚刚换了工作，因此最近谈论了许多关于为什么我决定这样做的事情，这并不令我惊讶。大概当某人离职时，不管是同事、经理、人力资源的人、朋友和家 人都很想知道为什么。从我个人来讲，针对这个问题，我倾向于给出一个不那么令人满意的答案，比如“我需要一个职业发展的更好的机会”，或者“我就是觉得是 该改变的时候了”。这就是企业版的“不是因为你——是因为我。”当我给出这样的答案时，我并不是在耍手腕或者回避。给出这样的答案是因为，我也不知道正确的答案。



要误解我。不管你到哪里（或者离开哪里），总是有一大堆抱怨或者愤怒，而总会有一些人站出来说：“你觉得这样怎么样，在家穿着睡衣就能挣到两倍的钱，并 且做你能想象的最酷的工作？”或者你现在的老板会说：“我们会把你的薪水减半，强迫你做一些糟糕的枯燥乏味的工作，并且把你送到南极洲去做这些。”我有一 个确切的离开的理由，但是那相对于常规来讲是个例外，这很有可能。

作为一个平常习惯，我喜欢考察我做事的动机。我认为这是一个很好的检查，以便确保我现在神志清醒，而不是冲动或者幼稚。因此，我把这个习惯用到我对于下一 步的决定，而结果如本文所述。请注意，这是对于是什么让我思考出这些文字的一个引用解释；我把我对于自己处境的观点，概括为一个软件开发这的大池子。然 而，我并不是要说：“我是最棒的，而这就是某人怎么留住我的。”我觉得，我的编程天赋层次，与本文无关；我更愿意把自己想象成一个能干而又高产的开发者， 因为学习的热情而脱颖而出，并为我的工作而自豪。我不把我自己看成“摇滚巨星”，而我总是把这种妄自尊大的自我评价视作非高产、病态的。

别人怎么想

我在过去几年在我喜欢的博客里读过一些的关于开发人员流动率主题的帖子，我认为这些帖子都为这个主题提供了一个绝妙的背景。我将列出的最早的一个是Bruce Webster写的，叫“大脑神经系统危机：死海效应”，他为我们无论是在意识和潜意识层次模模糊糊地知道的现象创造了一个很好的术语。“死海效应”是对一些组织倾向于太关注保留而无意中保留了平庸的人才的同时却流失了更好的人才的描述：

到底发生了什么使得更有才华和效率的IT工程师是最可能离开的那些人 –如果你愿意也可以说消逝。他们是最不能容忍那些困扰大型组织的频繁做糊涂事和工作环境问题的人；他们也是最有可能有其他机会离开的人。

往往留下的都是“残渣”---没有才华和效率的IT工程师。他们往往是感激于他们拥有的那份工作，对管理没有太多的诉求；即使他们发现工作环境不如意，却 依然是最不可能在其他地方找另一份工作的人。他们倾向于巩固自己，成为关键系统的维护专家，承担那些没有人愿意承担的责任，这样组织就无法让他们离开了。

Bruce描述了一个范式来说明有才华的人离开的原因经常是他们厌倦了那些在权威岗位（默认情况下）上却没有才华的人让他们做荒唐事。在精英管理的社会里里可以发现有一个关于等级秩序的真实反例， 这导致在一个功能失调的情况下有才华的人要么尽量避免要么尽快逃离。

Bruce的文章主要是从组织机构的角度阐述的；他提到为什么许多机构最后充斥着一群平庸的“高级开发人员”、负责人和经理人员，他们对有才能者因为上面 种种而离开表现得无动于衷。来自Daily WTF的Alex Papadimoulis更详细地阐述了有才能者离开的原因：

几乎任何一个工种都有一个员工带给公司的综合价值峰值（生产力与成本的百分比）。我把这个峰值称作Value Apex（价值顶点）。 在最初，员工的有效价值为0。但是随着他对新环境的逐渐熟悉，他开始将他的技能和以往的经验应用到工作中，由此他的价值迅速攀升。在该员工逐渐掌握业务领 域知识和与同事分享自己观点的同时，价值将持续以指数形式增长。

然而，一旦这个员工分享完了他的所有外部知识，学会了业务有关的所有东西，并且应用了他的所有经验，这种增长也就停止了。在那职位上的这个员工已经达到了他的极限。他到达了价值顶点。

如果这个员工继续在那个职位上工作，他的价值将开始减少。曾经的“我们现在无法实现的新想法”变成了“一成不变的让人厌倦的陈旧建议，并且我们决不会那么做”。要么是以前的解决方案被草率地接受：“是的，我们以前做过这种项目”，又或者是 某个想法被粗暴地拒绝，理由是“那是五年前的事了，我们都听过那个故事”。这将导致自我实现的缺失，最终会逐步瓦解工作的动力（或激情）。

有经验的开发者明白这点。越过价值顶点常常引发一种“是时候离开”的内在感觉，并且在此之后不久将不可避免导致对当前工作的愤恨和厌恶。没有任何东西（即使是一整支女按摩师团队）能够缓解这种失落情绪。

另一方面，平庸者却有着稍微不同的价值变化曲线：价值收敛（Value Convergence，可以理解为价值趋于平衡）。他们最终适应了平庸的职位并且一直呆在那里。他们的价值没有下降的唯一原因归功于他们积累和创造的巨大数目的机构知识。

这与单纯的精英本末倒置而导致有本事的开发者离开不大一样，而且更有趣。Alex的解释是，顶级的程序员，只有在提供给他们价值、或者他们能够提供使其价 值持续增长的工作中，才会快乐。最优秀、最明智的人，不仅想要成长，同时还想感到他们正在变得越来越有价值并成为有价值的代名词，我的理解，这就是一个人 工作中的自豪感。■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201306/398748.htm



有人问我，是做底层技术，还是做业务。这问题让我思考了很多，不由自主地回顾了一下我这十多年的软件开发经历.....

十年经历：软件开发的“三重门”

前言

这篇文章必然是通过我的个人经历来写的。所以，我先说说个人经历吧。我的经历基本分成三个阶段。

第一阶段：我刚毕业时在家乡的某银行工作，做些银行的业务系统，还搞些网络，电子邮件系统，OA什么的，因为大四的时候在老师的公司里实习，银行里的人际关系太复杂， 而且技术都包给了厂商，所以在银行的每一天都觉得不能适应里面的工作环境。两年后离职，单位分的房也不要了，直接去了上海，在上海呆了两年，本来想做互联网的，但是泡沫来了，最终去了一家做系统集成的国企公司还是继续做银行业务。这四年来，主要解决的都是一些业务上的问题，银行里的会计业务，OA业务，国 际业务，中间对公业务都非常地复杂，而且因为当时的软件开发相当的不规范，所以基本上是在一种比较混乱的状态下度过的，而银行方面又很强势，所以，这段时 间主要是做业务。所以，技术上主要是积累了如何使用那些技术。C+/Java， Windows编程，Unix编程，网络编程主要是这段时间学的，看了太多的书（我大学课程里没有C++和Java，也没有Windows/Unix和网络编程，所以，只能拼命地看书和自学）。

第二阶段：然后，我来了北京，到了一家做分布式计算系统的公司，整天和一个高性能技术高可用性的企业级的集群式的软件产品打交道（这家公司去年被IBM收购了），在这家公司把Windows/Unix和网络编程有 了更深入的了解，对我长进比较大的是明白了怎么做一个性能高，可用性高的集群式的系统，天天和底层打交道，干了4年多。然后去了一家金融信息公司，这家金 融公司主要做全球的金融信息数据处理，而我主要还是做核心数据发布系统的性能调优的项目，金融数据的实时性要求的高，数据量非常地大，高可用性要求得高， 得想尽一切办法省网络带宽，增加系统性能，还要保持高的可用性，不当机，不丢包。又干了4年多，入职的时候从国外接过来两个系统，其性能单机每秒可处理 120K message，我走的时候，我和团队把其优化到了每秒1.4M messages 的吞吐，另一个系统，从接手时的100k message/s优化到了500k message/s。这八年多的时候，全是在和这些高计算高性能的项目打交量，几乎没有什么业务，都是纯技术，积累到了很多和性能有关的高并发高计算系统 架构级的知识。

三阶段：两 年前来到了现在的做电子商务的互联网公司，还是在做一个数据处理量很大的业务系统，因为要干的是要把电子商务全球化的东西。但是，因为电子商 务的特殊性，必需要去兼顾业务的

特点，而且在这家电商公司，耳读目染了很多有趣的业务难题，比如，库存计划，配送优化，等等。虽然很多东西还不明白，但发 现，用技术来解决业务难题真是太有意思了。

我的这三个阶段，第一个阶段花了4年，第二个阶段花了8年，第三阶段刚刚开始2年不到，有时候我也去别的公司讲课，所以，我很有幸经历了中国软件开发的进化过程。我的经历可以说是中国软件行业进程的一个缩影，而我把这三个阶段称为——软件开发的三重门。它们分别是：

业务功能　粗放地开垦

业务性能　扩大化生产

业务智能　精耕细作

之所以加上“业务”二字，是因为我以为计算机是一个工具，其用来解决实际问题，所以，什么都离不开业务，就算是性能优化也一样，通过之前那篇“12306.cn的性能优化”中的“业务分析”段落，我们可以知道业务的不同，系统的难度和解决方法就可以不同。所以，我们总是用技术在解决业务问题。业务的形态对软件的开发有决定性的作用。

下面让我具体描述一下。

一重门：业务功能

这 是软件开发的第一重门，也就是掌握可以实现业务功能的技术。通常分成三块：语言+系统+数据处理。在这个阶段，主要是能掌握各种技术，比如：开发用的各种工具（如：IDE，XUnit，Debugger，等），各种代码库和框架（如：C++的STL，ACE，Boost，等，Java的Spring，Hibernate等），各种系统知识（如：Windows API，Unix/Linux API，TCP/IP，Socket，多线程多进程间的同步、互斥，并发安全，还包括Web平台，移动平台，等等），还需要掌握数据处理的知识（如：数据 结构，基本算法，数据库设计，数据库引擎，SQL等），等等……

这个阶段主要是把这些不同的技术组织成可以实现业务功能的解决方案。重点是能掌握和使用技术。很多流程和方法论的东西基本上就在这一重门里。这重门主要解决的是业务实现问题。

二重门：业务性能

业务的功能搞定了以后，就是业务的性能问题了。搞定功能并不难，搞定性能是有点技术含量的事。有句话不是那么说的吗——每个人都可以搞一个网站出来，但不是每个人都能搞出能支持百万级访问量的网站。但是，我看到很多技术团队或是工程师脱离了业务，只单纯地搞性能，比如：单台服务器支持10万个TCP链接的并发，等等。这些东西虽然在技术上有点意思，但是没有业务的环境，也只能是自娱自乐了。

我们可以看到一些企业开始注重这个问题了，性能问题也是最近被大家讨论得最多的问题，京东商场的性能问题，12306的性能问题，等等。

当然，所谓性能不并单单指系统的吞吐力，还指系统运行时的总体性能，比如，系统安全性能，易用性能，系统的Accessbility的性能，系统的扩展性性能，等等，就像是前段时间“Web开发中需要注意的问题”一文中谈到的那些事一样。这表明着你对系统的全面和深入的了解。

在这个阶段，需要对业务模型，数据流，业务流，系统架构，算法，和各种技术有深入的了解，要了解到本质上来。比如，在第一重门中■

本文未完，请点击下面连接阅读全文：
http://developer.51cto.com/art/201306/399068.htm

偷了世界的程序员

■ 本中的四个程序员可能并不是那么声名显赫，而且也很不老实，或许算不上成功，不过他们的确改变了世界。本文有分析了互联网上P2P的那些事，相当有参考价值。

十年前，有四个年轻人改变了这个世界的运作方式。他们使用的并不是法律或是武器或是金钱，而是使用软件来改变世界。他们当时有着激进和极具破坏性的 想法，并把这些想法付诸于代码，在Internet上以免费自由方式发布。这四个人，没有一个完成了大学学业，却奠定了今天我们习惯的数字媒体环境的基 础。然后，因为各种原因，他们也迅速地消失在公众视野中。

1999年，美国东北大学的一个叫Shawn Fanning的一年级新生开发Napster，从此，成为了P2P文件共享和不需要大型机构或零售商就可以获得音乐的先锋和范例。《时代周刊》和《财富》把他放上了封面。那时，他在19岁。

就在同一年，一个挪威的只有十几岁的年轻人Jon Lech Johansen，他和另两个今天都不为人知的程序员，写下了一个程序解密了商业的DVD，而他成为了全球盛名的“ DVD Jon.”，那年，他只有15岁。

而在1997年，Justin Frankel，一个亚利桑那州塞多纳的18岁的黑客，开发了一个免费的MP3播放器——WinAmp，其成为了Windows操作系统上装机必备的软 件，并造就了主流数字音乐的革命。在他发布的第18个月内，1500万人下载了这个软件。而三年后，Frankel 开发了 Gnutella，一个P2P的文件共享协议，没有中心结点，不像 Napster，其不可能被关闭。目前有上百万人还在使用它。

2001年，Bram Cohen, 当年 26 岁，开发了一个P2P的文件传输共享协议—— BitTorrent，其以全新一流的架构全面优化了网络上大文件的共享和传输效率。 BitTorrent 也变成了整个Internet上发布大数据和文件的一个标准。

在 2000年代的上半段，《时代》采访了这四个程序员。那个时候，看起来他们要以数字化动乱把整个复杂的传统媒体娱乐平台给拆除，而对有版权的电影，音乐和 电视的收费则变得困难和不可能，那些艺术家也将无法从他们作品得到报酬，整个娱乐业包括时代华纳也将被炸为平地。而盗版业则借着这四个程员的软件侵袭了美 国公司。



毕竟”，我们在2003年报道到：“在整个信息经济中，不可能所有的信息都是免费的”。如果毁灭正在来临，那么， Fanning, Johansen, Frankel 和 Cohen 将是那“四骑士”（译注：启示录中的四骑

士传统上被解释为瘟疫、战争、饥荒和死亡）。

没有毁灭

毁灭并没有发生。但是整个娱乐业因此而改变，而这些改变的复杂性和逐渐演进超出了我们的期望。这些发生的故事， 海盗王们的事，对于今天数字化世界正在发生的事情有非常高的参考和教育价值。Fanning, Johansen, Frankel 和 Cohen 现在都硅谷运作着自己的小的，合法的软件公司。他们现在没有在做和盗版有干系的事情——当然，如果他们真的没有。

Fanning，四个人中唯一一个没有回复我们的采访请求的人，他较早地退出了毁灭传统唱片业的事业。在2001年，Napster因为不堪众多关 于其协助并煽动版权侵权的法律诉论的重压，而不得不关闭。2002年，Fanning 创办了新的服务 Snocap ——他尝试把文件共享合法化，在和相关的唱片公司合作下，Snocap 赋予消费者对其下载作品给于创作者报酬的权利。

但是，到那个时候，免费自由的文件共享程序像病毒一样的增涨，而用户则热衷于更换他们的音乐硬盘。他们仅在2001年8月一个月内就交换了30亿个文件。而要从这些文件交易中收到钱是根本不可能的。是的，要和免费竞争是很难的一件事。 Fanning 创造了一个连他自己都搞不定的怪物。

所以，他停止继续尝试Snocap下去。 Fanning 的下一个项目是给游戏玩家的社交网络叫 Rupture，最终，他在2008年时以1500万美金把其卖给了电子艺界Electronic Arts ——这是他的第一次发薪日。他现在又于2008年11月开了一个公司 Path， 其主要提供给iPhone手机进行照片分享的服务。

而Napster呢？今天他还在。这个商标在破产拍卖时被卖了，然后再被卖了，但其再也没有被 重建。现在其被 Best Buy 运营，其是 iTunes 的竞争者，其口号是—— “More than just a music store.”（不仅仅只是音乐商店）

没有盗版的人

作 为 Gnutella 的作者， Justin Frankel 是 Fanning 合法的继任者。不像 Fanning，他很早就收获了他的第一桶金。在1999年，当WinAmp大放光芒的时候，AOL买了WinAmp和他的公司——Nullsoft，价格应该在1亿美金左右。这让 Frankel 在20岁的时候就非常富有。当然，他也成了AOL的员工。

但这并不是很匹配，在Nullsoft, Frankel的做法是把软件开发到极致，然后免费发布出去。而在AOL，软件的商业销售威胁并压倒了软件本身。“我致力于的产品，就像这样，我们不愿意金钱的掺入，我们正和其它公司做这笔交易，所以，产品也只能是这样 的结果”，他回忆到，“没有人真正地去关心用户的体验是怎么样的”。

与此同时，Frankel 用他的业余时间开发 Gnutella 。这是一个很有才的软件，不像Napster，其是真正的分布式，没有中心服务器，这样，也没有那个“关闭按钮”让那些律师按。在2000年3月的时 候，Gnutella上线，其发了一个贴子：“看见没？AOL也能给你一些好的东西！”，但是就算是这样，也没有换来AOL对其忠爱，而一大堆互联网公司 在那时试图并入大的媒体公司，在Napster被诉讼的中期，2004年，他离开了AOL。

然后，他开始干了些有趣的事：他离开了他的成功地，他不用Gnutella，没有花一毛钱，就算是10年以后也是这样。■

本文未完，请阅读原文：
http://developer.51cto.com/art/201306/398259.htm

编者按

内容仅作学习用途，版权归原作者所有。本书简介：由GAMESbrief出版（http://www.gamesbrief.com），伦敦，2012年10月。作者：Nicholas Lovell & Rob Fahey

免费游戏设计法则：系列简介

主要内容目录（暂译）

第一篇：设计法则

法则1：开心最重要

法则2：星巴克测试

法则3：来了就不想走

法则4：层级复杂性

法则5：永无休止的重要性

法则6：大方点

法则7：永远免费

法则8：1美金只是个开始

法则9：花100美金不算什么

法则10：注重奖励而非虚有其表

法则11：新手教程别烦人

法则12：主角可不想挂掉

法则13：情感驱动，而非内容

法则14：体验和学习

法则15：游戏开发永无止境

第二篇：设计免费游戏要知道的一些事

A/B测试

用户获取成本

约定机制

ARPPU

Bartle模型

转化率

货币化

GAMESbrief 免费游戏表格

操作性条件反射

冥律

斯金纳盒

鲸鱼用户和真正的粉丝

前言

为什么我们需要了解设计免费游戏的法则？

答案很简单，“免费”这个概念威力无穷但又令人惊恐。在如今这个年代，游戏开发中不得不习惯于开发免费游戏。数字通讯工具从很多方面改变了我们的生活。但是对创意人员来说，最大的剧变莫过于数字媒体产品（音乐，书籍，游戏，电影）发行渠道的

成本已经接近于0.之前在消费者和产品之间有一个巨大的鸿沟，而如今天堑变通途。

没有人可以在洪水到来之时傻站在那里不动，还要求洪水迅速退去。我们必须必须正视变化，并对变化做出正确的反应，而不是坐以待毙。

而应对这一变化的重大举措之一就是设计免费产品。与其抱怨传统的对用户收取费用来出售产品的方式已经日薄西山，还不如采取有力的措施。如今我们可以直面最大的用户渠道，还可以随时了解他们的整个游戏体验。对于游戏开发者来说，客户关系管理变得无比重要，开发者应把玩家当做独立的个体来看待。

免费游戏的理念无比强大，但创意开发人员和游戏公司还需要转变自己的一些观点。10年前，在美国如果说90%的玩家免费玩你的游戏，你首先想到的就是游戏被盗版了。而如今，这一数字说明有10%的付费转化率，已经是相当不错的成绩了。如果这还不足以为奇的话，那么想想另外一个事实吧，这10%的消费只是花在简单的游戏内道具和货币中。而你耗费苦心创造的整个游戏世界只不过是提供了一个进入的途径而已。游戏开发者的设计理念和之前相比应该完全颠倒过来。

这就是为神马我们需要了解设计免费游戏的法则。尽管这些法则看起来似乎就是大白话，就像心理学家经常通过无数次的实验告诉我们一些基本上人人都懂的事实，比如婴儿更喜欢盯着美女看，诸如此类的一样。但要真正的转换到免费游戏的设计理念，开发者需要重新审视游戏开发和设计的很多基本“真理”。本书所介绍的法则来自行业中一些最优秀的免费游戏开发者，他们所提供的这些概念和方法将有助于每个游戏开发者重新认识，理解和思考游戏的设计方式。

不过且慢，所谓的法则不就是让人打破的吗？

当然，这的确是个事实，任何法则都是如此，但这也是我们需要法则的另一个原因。为神马一个英语考试不及格学生的拼写和语法错误会被老湿毫不客气的画上红叉，而同样看起来内容支零破碎且不符合语法习惯的James Joyce的尤利西斯，还有Jonathan Safra Foer的真相大白却被认为是惊天地泣鬼神的大师神作？难道仅仅是因为他们的身份吗？当然不是。大师之所以敢于打破法则的约束，恰恰是因为他们对法则已经足够的了解，其了解程度要远远高于大多数的普通作家。所以在学习任何新东西的时候，都有一个归零心态，先不要对法则不屑一顾，stay foolish,stay hungry。先从零接受，然后逐渐理解，到完全掌握，然后就可以不受法则的约束了。想一步成为牛人而不作出辛勤的努力工作其实是懒惰的思维。

当然，换句话说，当你已经达到或超过一定水准的时候，可以不必考虑法则，甚至任意打破法则。但在你完全理解并真正在实践中掌握这些法则之前，最好还是老老实实的多学点东西，没坏处。

好吧，一大堆废话。这就是为神马我们需要法则，以及需要真正理解法则的原因。学习法则的目的就是为了有一天可以完全突破法则的限制。

Nicholas Lovell是GAMESbrief的创始人，为Square Enix, Channel 4,IPC Media, nDreames, Firefly, Atari, Rebellion 和Channelflip均做过免费游戏业务方面的咨询。他常年在gamasutra,Develop,mCV和华尔街日报上发表关于数字媒体方面的文章。他同时是The Curve的作者。■

本系列相关文章，请访问
http://developer.51cto.com/art/201305/395200.htm

我们“看不到”的残疾人程序员

残这个词在中国贬义大于褒义，对于正常人来说就是包袱、累赘的代名词。中国是一个不适合残疾人居住的国家，在这里导盲犬上公交车有时是禁止的，盲道上是停着各种车辆的，残疾人想考大学是非常困难的。我们今天要来看看那些我们“看不到”的残疾开发者。

公开“作秀”找工作的张肖敏

一场结膜性脑膜炎夺去了永泰山区小伙张肖敏健康的双腿，因为残疾他不得不从辽宁省交通高等专科学校退学。由此他失去了所谓的第一学历，只能透过3年的时间拿到集美大学会计专业专科文凭，并最终拿到计算机信息管理专业本科的自考文凭。



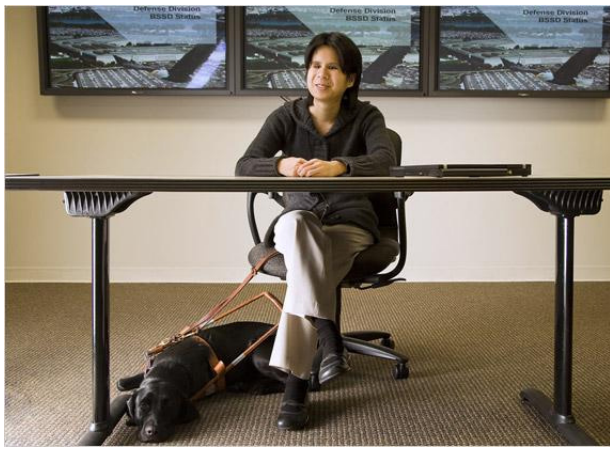
身残志坚张肖敏

当他需要13万在北京宣武医院进行背骨髓电刺激手术的时候，所谓的医保或者大病医疗保险对他是不适用的。那么找一份IT相关工作，靠自己双手去劳动、生活、治病成为这位PHP程序员最大追求。

2013年被成为史上最难就业季，700万拿着“全日制本科”烫金字文凭的大学毕业生正在到处碰壁找工作。而张肖敏这样只有第二学历的自考生，想必找工作就更加难上加难了。或许在IT公司楼下竖起招牌，在微博上请大家帮忙扩散是显得高调了一些，不这么做，有人会录用他么？

盲人程序员是怎样炼成的

毕业于德州大学奥斯汀分校计算机科学系的张海伦，现在为一家联邦政府合同商工作，负责为美国国防部开发定制系统软件和网站。



作为盲人，如何“看”计算机屏幕就成为最大困难。这里借助读屏器、光学字符识别、电子盲文记事本以及盲文触摸屏等工具，弱视者利用操作系统内嵌的放大工具或插件，完全可以在Windows、Mac和Linux等系统平台上，熟练地使用C/C++、Java等各种编程语言编写程序，制作网页。一些原本对盲人程序员不太“友好”的语言环境，例如python等，也在改进它们的代码程式，剔除妨碍盲人程序

员读写的某些障碍。

残疾程序员遭遇不公

在某开发社区的pksniq，因为公司的不公平待遇而愤然离职。渴望与正常人一样生活、工作，是每个残疾人的心声。

今天我向经理提出了辞职！来到公司已经一年多了，每天都无事可做，幸亏自己自学能力强，否则这一年就虚度了。回想这一年，真的有许多无奈，残疾人程序员活的很悲哀，别人总以为你没法找到工作，以为给你一个工作机会就不错了！我想说，给我工作机会我很感谢，但不能把我们残疾程序员当孙子使，我敢说，讲技术我可以，做项目一般人不如我，我不怕辞职后不好找工作，别以为残疾程序员就一定在你那干一辈子，怕被你开除！所以今天我提出了辞职，我要证明自己！

——pksniq

作为沈阳城里的一名程序员，当他同学第一年拿到2000+工资的时候，他最多只能拿到1850，这还是在以前1600的基础上有了一定程度的上涨。同工不同酬，造成的不仅仅是生活上的不公平，更是对残疾人心理的摧残。

我们怕残疾人什么?

在普通中国人眼里，残疾人的形象大抵有两种。一种是缺胳膊少腿，上下楼需要人搀扶，甚至需要专门请人喂饭，端屎端尿的“废人”。另一种是患有脑疾病，目光呆滞毫无生气的残疾人。

我们怕看到他们残缺的身体，怕他们在地铁里扯着我们衣角乞讨。所以才有人感叹，在中国，残疾人显得尤其少。因为那8500多万中国残疾人只能窝在家里，躲避别人的眼光，了此残生。

北京为了2008年残奥会，才开始重视残疾人设施的建设。银行门口的轮椅坡道，公交车上的残疾人专座才慢慢出现在人们的视野中。中国残疾人，特别是中国残疾人开发者只有等到全社会把他们当做正常人一样对待之后，才有可能迎来自己的春天。

这条路，任重而道远！■

专题推荐



导读：2013年6月苹果WWDC大会发布了iOS 7，其颠覆性的扁平化界面让业界感到拟物化设计风格已经风光不在。当苹果、微软都投奔扁平化风格之后，未来的WEB开发者应该对这种被“拍扁”的UI世界有更多的了解。

点击【链接】进入专题



导读：Disruptor是一个高性能的异步处理框架，或者可以认为是最快的消息框架，也可以认为是一个观察者模式实现，或者事件-监听模式的实现，直接称disruptor模式。disruptor最大特点是高性能，其LMAX架构可以获得每秒6百万订单.....

点击【链接】进入专题

“保卫妞妞”一位WEB工程师的手游初创之路

■ 朱念洋，腾讯开放平台部软件工程师，一位拥有5年以上开发经验的WEB开发者。是什么让他有了开发“保卫妞妞”这款手游的想法？51CTO近日专门采访了这位敢于

在移动互联网火热的今天，众多的创业机会和应用商店中那亮光闪闪的美元让很多人摩拳擦掌。相信很多互联网企业，甚至传统IT行业中的开发者已经开始琢磨怎么给自己挣得第一桶金。

“保卫妞妞”内容提要

一起来保卫妞妞，打倒怪兽吧! 只需要舞动你的手指就可以啦！ 游戏中有两种武器，分别是 刀和手。刀的操作方法是手指在屏幕上滑动，被切中的怪物就会受到伤害。

扫描二维码直接浏览下载页面

兴趣决定方向

从个人兴趣出发，是一个很好的方向。例如51CTO之前曾报道过的“可可私房菜”的开发者本身就是一个所谓的“吃货”。而开发“保卫妞妞”的朱念洋，自己也是一个手机游戏爱好者。他玩过的手游很多，类型也很多。比如“大掌门”，“武士大战僵尸”之类。不过感觉最喜欢的还是“水果忍者”，它和目前智能手机的触摸屏操作就结合得很好，而且游戏本身的设计也是可快可慢，张弛有度，玩起来很舒服。

玩的游戏多了，也就慢慢发现App Store游戏排行榜里全是网游，同质化过于严重。大家都出于商业化的目的做手游，让朱念洋想能不能先让用户觉得好玩，再考虑顺便赚点不算多的钱?

再着，从PC上移植的手游，甚至有些原生的手游，在部分操作上并没有完全考虑一些触屏用户的操作习惯。比如朱念洋就对有些游戏在屏幕上模拟一个摇杆感到不可接受，操作起来比较别扭。

基于上面的两点想法，本文主角朱念洋开始了自己的创业之旅，要做一款与别人不一样，充分考虑与触屏操作相结合的游戏。因此在他的手游作品“保卫妞妞”中很多动作，比如手指点、切、拽就是这样产生的。

万事开头难

对于朱念洋来说，最大的技术困难莫过于没有系统的学习过游戏开发。尽管朱念洋有5年左右的C、C++开发经验，3年左右的Python开发经验，多年WEB开发经验，但就是没有游戏开发经验。所以一开始的游戏架构是边写边改，浪费了不少时间，这也是整个开发过程中需要一直解决的问题。幸好最后的架构，朱念洋还是挺满意的。

Objective-C语言倒是没有难倒我们的开发者，语言本身想通的地方居多。Objective-C本身设计的很简炼，所以学起来并没有感觉到任何不适（除了它方括号的语法，不过习惯了也就没什么了）。学

会语法就1、2天吧，主要还是要在实战中多用才能熟练起来。

幸亏有了COCOS2D这样的游戏引擎，才让朱念洋的开发更加顺畅。一个WEB开发者如何驾驭COCOS2D，他推荐《Learn cocos2d on iPhone》这本书，不过书中用到的COCOS2D是0.99版本，而朱念洋游戏中用到的是2.X版本，所以很多代码不能用还是让他痛苦。

万事俱备，开始动手吧！

看别人写代码和自己实际开发完全是两回事，怎么检测是否受到攻击？怎么模拟碰撞？怎么加入敌人AI？还是完全不知道怎么下手。找个与自己设想一样类型的游戏作为教程应该就好多了，朱念洋还真就找到了www.raywenderlich.com这个网站。在这里很多都支持COCO2D 2.X版本，并还有支持中文的教程。这些技巧对朱念洋的开发有很大的帮助。

论程序员有个好老婆的重要性

在谈到“保卫妞妞”这款游戏的情景设置问题时，朱念洋觉得这个过程挺戏剧化的。来是打算以朱念洋老婆为原型来作游戏主角的，结果美工那边画出来之后一点都不像，但是又很可爱，所以就这么将错就错了。

主角设计的很可爱，而怪物是故意设计的有点恐怖的。这也是之前一个美工的朋友告诉朱念洋的：游戏中的对抗也要体现在美术上，如果怪物也用可爱的风格，那么整个游戏风格就会偏向低龄化，所以让主角和敌人的对比强烈一些，冲击力就会更好一些。这一点“植物大战僵尸”做的就非常好。

从朱念洋的这个例子就可以看出，一个好老婆可以带来多么大的灵感。所以，各位单身程序员，赶紧甩掉电脑和手纸，去相亲吧。

选择iOS的个人创业者

为什么选择iOS？这个问题肯定会有很多人要问朱念洋，他给出的答案很简单“因为一开始就打算用付费下载的盈利模式，所以选择了iOS。”至于Android和Windows Phone，暂时不打算在上面做游戏，不过他们一款交友应用就要在Android平台推出，欢迎大家下载试用。这将是一款WEB APP。

盈利是每个个人开发者无法避开的问题，中国国内对于正版的概念不太强，朱念洋的“保卫妞妞”先做英文版也是希望在国外获取一些收益。

朱念洋自己就是一个从兼职创业到辞职创业的探路者。他表示，不辞职创业的话，如果工作比较轻松，那么晚上来做自己的事情还是很爽的，可以没有任何风险的创业，但是也有些问题，比如有些会议因为竞业条款不能参加之类的。辞职之后这点顾虑就没有了，但是风险也提高了，赚不到钱明天就要饿肚子，算是有得有失吧。对此他的建议是，如果一开始是不辞职创业，那么等到能够赚到钱的时候再辞职创业，风险会低很多。他有朋友也在兼职做手游，也是待到时机成熟才会有更大的动作。

对于变现，朱念洋倒不是很在意。他最希望的是丰富一些关卡，增加更多的敌人和道具，让玩家玩得更开心。

朱念洋30天手游创业过程中，一共投入了5天学习，20天的开发时间。前后的投资大概3000元人民币，这主要是美工设计方面的成本。如果前期计划得更合理些，20天就可以完成大致的开发。

在手游行业巨头涌入的今天，已经涌现很多个人手游开发者将被淘汰的言论。从朱念洋的例子中我们可以看到，兴趣和决心才是个人手游开发者做好的前提条件。各位程序员，你是否已经下定决心闯出自己的一片天？■

MySQL创始人：毫无理由再用MySQL

MariaDB不仅成为主要开放源码操作系统的默认数据库，更在 WikiMedia 等重要组织与企业，取代 MySQL 的地位。日前 SkySQL 与 MariaDB 的总公司 Monty Program Ab 进行合并，扩编了开发者团队。以下是 Monty 日前进行的专访，谈到了数据库领域的改变。

问：MySQL 是全球主要项目与企业的默认数据库，MySQL 如此成功的原因为何？它有哪些特质？是其开放源码的特性、技术优势、或是 Sun 的营销能力？

Monty：MySQL 在被 Sun 买下前就已经很普及了，这也是 Sun 之所以买下它的理由之一。

MySQL 之所以受到如此欢迎的原因是：

1.自由软件／开放源码

2.我们从一开始就对用户非常友善并且乐意提供协助

3.前五年我个人为了协助用户就写了 3 万封电邮

4.双重授权

5.让我们赚到钱好雇用更多人加入项目

6.让我们吸引到投资者

7.从一开始释出就在大量平台上测试 MySQL 二进制文件

8.当我们开始 MySQL 时，多数自由软件仍仅有原始码，也没有对所有平台做测试

9.从一开始我们就设定好，要让所有人都可以在 15 分钟内取得并执行 MySQL

10.我们很幸运在对的时机推出对的产品

11.当我们开始时，市场没有自由 SQL 数据库的竞争者

问：两大开放源码势力都出自于芬兰人之手，也就是 Torvalds 与 Widenius。芬兰的教育系统是否鼓励这些行动？

Monty：在这两个案例中，教育系统对产品并无太多影响。

我想比较多是出自于芬兰人的本性，我们很固执又讨厌放弃，即使其他人都说我们做的事一点用都没有，而且我们不可能与大公司竞争。

就 Linus Torvalds 与我而言，我们都有一种天分，让和我们谈过的开发者，对我们所做的事感到兴奋。我们两个人的母语都不是英语，也许这一点有助于和世界各地的开发者进行沟通，因为从一开始我们不设限加入项目的人必须会说流利英语。

问：你曾对 MySQL 卖给 Oracle 有所疑虑（你的疑虑也成真了）。MySQL 的普及率最近正在下滑。你认为下滑的原因为何？Oracle 做错了什么？

Monty：是的，MySQL 近来没落了，不过它所流失的用户群，多半都跑向 MySQL 生态系的其他产品，像是 MariaDB。

Oracle 做错的事有：

1.MySQL 新的企业延伸套件采取封闭源码

2.数据库的臭虫不再公开

3.不再积极更新 MySQL 的公开代码库

4.未能快速传达或修复安全问题，这一点让 Linux 散布套件对 Oracle 很不满

5.Oracle 选择移除功能，而非修正臭虫

6.MySQL 5.5 的新程序代码不再有测试案例

7.来自 Oracle 的部分新程序代码好的出乎意料，但不幸的是质量参差不齐，相当可观的一部分在我们纳入 MariaDB 前，更需要重新改写

另外，或许最糟糕的是，社群没办法与 Oracle 的 MySQL 开发者合作。因为：

1.Oracle 不接受程序修补

2.没有公开的发展蓝图

3.没有任何方法可以和 MySQL 开发者讨论实作细节以及目前程序代码的运作

问：在此同时 MariaDB 逐渐受到重视，对一般用户而言，MariaDB 与 MySQL 的差异为何？

简单来说：

1.MariaDB 比 MySQL 多了 30 个人年以上的开发工作量

2.更多功能

3.更快

4.更稳定

5.安全问题更少

最简单的答案是，毫无理由不选择 MariaDB 5.5 来取代 MySQL 5.5，这个答案下一版本依然适用。

问：MariaDB 如何改变数据库领域？MySQL 仍在 Web 上占有主导地位，有任何鼓励网络托管服务供货商改用 MariaDB 的计划吗？

Monty：有许多网络托管服务供货商已经支持 MariaDB。你只需提出询问，许多供货商就会提供 MariaDB。

我们正在积极游说网络托管服务供货商，请他们公开支持 MariaDB。这的确是个耗时的工作。

问：MariaDB 可以用来直接取代 MySQL。撇开双方在技术上的相似性不谈，这两个项目在开发与授权上的主要差异为何？

Monty：MariaDB 是由具有开放发展蓝图的社群所开发，任何人都可以轻易参与其中。MySQL 被发展成一个封闭源码的项目，只有 Oracle 和少数 Oracle 选出的公司才能参与。MariaDB 和 MySQL 一样采取 GPL，不过 MariaDB 有给 C 和 Java 用的 LGPL 驱动程序。我们也在开发 LGPL ODBC 驱动程序。如果你使用了这些驱动程序，在大多数情况下，你并不需要购买 MySQL/MariaDB 的授权。

问：从 MySQL 转移至 MariaDB 的难易度？

Monty：只是小事一桩。只要移除 MySQL，然后安装 MariaDB。你的所有工具，连接程序都可以如常运作。你也不需要导出和汇入数据。格式与文件名都是相同的。

问：在 WikiMedia 基金会等组织，或是 openSUSE 与 Fedora 这些 GNU/Linux 散布套件上，我们已经看到若干采用 MariaDB 的例子，但透过 Monty Program Ab 进行商业部署呢？我的意思是，你公司的营收现况为何？

Monty：Monty Program Ab 从某些公司取得了 MariaDB 的开发资金。此外，Monty Program Ab 也从提供第三级支持给 SkySQL 等公司而获得资金。■

VS 2013在ALM上的改进

ALM（application lifecycle management）应用程序生命周期管理

在近期召开的北美TechED 2013上，Visual Studio 2013预览版已经发布。本文将谈到微软负责软件开发者部门的全球高级副总裁S. Somasegar所关注的Visual Studio 2013在软件生命周期管理上的众多新功能。自从Visual Studio 2012发布以来，Visual Studio 2012中的下载量已经超过400万次，在过去任何Visual Studio版本采用最快。通过两个VS的更新，VS2012.1和VS2012.2，对60%正在使用的Visual Studio 2012部署上了新的更新。S. Somasegar也已经看到Visual Studio 2012提供了新的价值，在Visual Studio中可用的功能，进一步扩展这个强大的生态系统和集成的解决方案，其中包括市场上的近500个VSIP产品，和Visual Studio Gallery中3900多个产品和插件。

我们（原文作者：S. Somasegar ）不仅看到通过在客户端上的机会。在云中，我们在Build 2012上发布Team Foundation的服务，已经更新了新的功能。大约每三个星期就提供新的功能，比如2013年1月份的ALM峰会上我们提供对Git的支持。

Visual Studio有这么多先进的功能，能帮助开发人员和团队完成现代应用程序生命周期管理的建设。随着建立、测量、学习周期的缩短，开发团队能够在开发过程中与其他利益相关更容易协同合作，支持连续的发布节奏，平衡敏捷与质量，消除传统开发经营者与企业利益相关者分离的现象，推动应用与业务成果之间的链接。微软在Visual Studio 2012及其更新中正在扩展ALM功能，进一步提升工具和服务的质量。在DevOps的背景下产生摩擦更少，质量的生产路径。现在，在北美TechED 2013上，我们正式宣布与应用程序生命周期相关的几个关键进步。

Visual Studio 2013

我们很高兴分享下一个版本的Visual Studio 2013，将会在今年晚些时候的旧金山Build 2013发布一个预览版。在TechED 2013上，布莱恩　哈利已经提到一些新的ALM特性。比如专注于业务的敏捷性，质量可控和DevOps新功能。下面是我最喜欢部分：

1.敏捷组合管理，这可以让你“大规模”的计划你的敏捷项目，让整个组织的多个团队之间的层次关系更加明晰。

2.基于云计算的负载测试，一个新的Team Foundation服务能力，充分利用Windows Azure的弹性可扩展性产生的流量，模拟成千上万的并发虚拟用户，以帮助你了解你的Web应用程序和服务的负载能力。

3.代码信息提供单元测试，工作项，代码引用，以及更多。直接在Visual Studio代码编辑器内显示信息以提高开发人员的生产力，使开发者无需离开编辑器中就能查看项目相关的上下文信息和指标。

4.为TFS创建一个团队空间，这个TFS集成的聊天室将提高合作团队成员之间的数据分享和其他相互作用。

5.Visual Studio集成身份识别，这样的IDE就可以

获得后端服务的支持，例如，开发者在不同安装设置中进行切换。

6.TFS将支持代码审查意见回返，提高代码透明度和可追溯性。

7.一个.NET内存转储分析器，它使开发人员能够轻松地检索.NET对象的内存转储，并比较和修复两个内存转储。

8.Visual Studio 2013中将内置支持GIt，无论是客户端还是服务器。包括在内部部署的Team Foundation Server 2013。

这些只是少数的新功能，在未来数周我们将讨论更多的新功能。这些新功能将在Team Foudation服务中提供。

关于InRelease

DevOps的应用程序生命周期管理是一个越来越重要的组成部分，面对越来越大的需求。因为企业需要开发和部署高质量的应用程序，以更快的步伐。我们将继续投资在完善现代应用程序生命周期，特别侧重于DevOps。

为此微软收购INCYCLE的InRelease部门，以提供先进的.NET和Windows Server应用程序管理解决方案。

MSDN和在Windows Azure上进行开发和测试

我们正在做的技术改进在Visual Studio上只是一个方面。我们正在努力提高生产率和让更多成功团队使用微软平台。

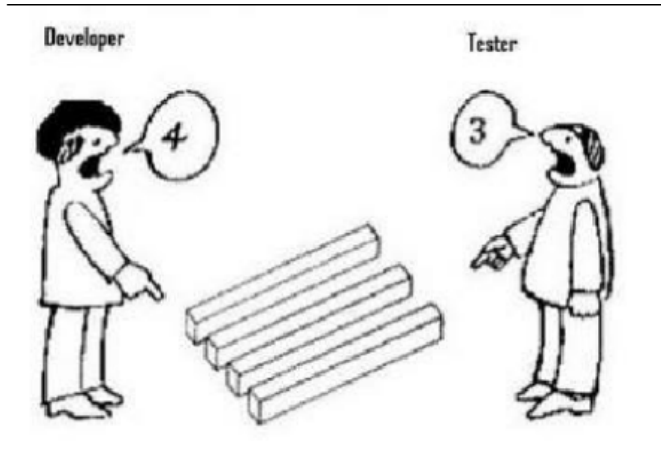
例如，我们已经改善了Windows Azure平台，将其提供给合资格MSDN订阅者。你现在有一个选择，如何使用你的Windows Azure开发和测试你的应用。无论你申请他们的虚拟机，网站，云服务，移动服务，媒体服务，HDInsight，或更多。 Windows Azure 对MSDN的好处包括预先与MSDN订阅软件，如SQL Server和BizTalk Server，虚拟机映像的访问，并可以将您自己的MSDN软件上传到自己的虚拟机。

此外，我们的目标之一是使它对于开发团队的每一个成员更加易用，无论是开发者或者是测试员。使用新的Windows Azure的MSDN开发和测试的好处是显而易见的。截至6月1日，MSDN订户可以使用权限来运行在Windows Azure虚拟机选定的软件（更多细节参见Visual Studio和MSDN白皮书）。

关于S. Somasegar



微软负责软件开发者部门的全球高级副总裁S. Somasegar。领导着.NET平台、Silverlight、Visual Studio、Expression Studio、IIS Web服务器、Media Server和Commerce Server等产品的开发。MSDN和TechNet也是他麾下的服务。 此外，他还负责微软印度开发中心和加拿大开发中心。■



看曹政如何减少SQL请求

首先为了防止某些专业挑刺人士无限制发挥，先声明几个前提

1：索引优化是基础工作，没做好这个其他的不用提，但本文不展开此内容。

2：优化数据库查询有非常多的分支，减少SQL请求只是其中一个领域，其他分支本文不涉及。

3：在部分场景下，甚至需要增加SQL以解决诸如分布式或其他问题，本文不涉及。

4：运维优化和其他优化手段本文不涉及。

5：产品业务逻辑优化本文不涉及。

6：其他本文没提到的内容欢迎自行联想，技术水准高超者请忽略本文。

第一 查询请求的分析和裁剪

线上系统，出现请求较多，数据压力较大（索引优化到位的前提下），我会让程序员输出一段时间的查询请求。（通常数据库操作有封装对象，直接记录日志即可，建议写入/dev/shm 以减少i/o压力，如果请求频次实在很高，可以取一定比例写入日志），然后基于日志分析。

1、完全一致的查询请求有多少，平均每秒会出现多少这样的查询。

比如常见的，所有页面都加载系统信息 select * from systeminfo;

2、基于同一数据表同一主键的查询有多少，平均每秒会出现多少这样的查询

比如 select name from userinfo where uid=10134; select email from userinfo where uid=10134;

这两种请求，是可以通过建立缓存机制来优化的，而且做了这个分析，会有很好的数据认知。

当前数据库每秒处理多少查询请求，其中可优化的冗余请求有多少，如果建立缓存可以减少多少请求。提升系统支撑多少?

对于一些不是特别出色的开源系统，分析一下会发现，可裁剪的查询请求是非常巨大的。

第二 更新请求的分析和裁剪

更新请求也可以优化，我们一般用mysql的情况下，是先解开binlog文件，还原为文本文件，然后分析基于同一数据表，同一主键的更新请求有多少，平均每个时间段出现多少这样的请求

举例1：update user set lastacttime=.... where uid=10314; 经常更新最后活跃时间

举例2：update posts set views=views+1 where pid=10004211; 更新同一个帖子显示数字

如果这样的请求较多，那么可以有针对性的建立队列，定时异步更新。

在异步更新过程中

范例1 多条请求只记住同一主键最后一条即可；

范例2 多条请求可以在程序中合并，对数据库操作只进行一次。

这样更新请求频次就极大下降了。如果线上有实时性要求，线上可以保持一个内存数据做同步更新。■

世界上公认的第一个网页是诞生于1992年，发展至今，网页已经有21年历史了。我们经常见到的Web音频就是这么样子的。现在，大部分的网页音频播放依靠着Flash，但是显然地，Flash已经越来越不适用于Web开发。所以呢，苹果放弃了Flash，而Android也宣布不再支持Glash。主要原因就是HTML5的诞生。

如何在网页中应用HTML 5音频

在HTML5标准网页里面，我们可以运用audio标签来完成我们对声音的调用及播放。以下是最经常见到的运用HTML5三种基本格式：

1.最少的代码

```
<audio src="song.ogg"
  controls="controls"></audio>
```

2.带有不兼容提醒的代码

```
<audio src="song.ogg"
  controls="controls">
Your browser does not support the audio
  tag.
</audio>
```

3.尽量兼容浏览器的写法

```
<audio controls="controls">
  <source src="song.ogg" type="audio/
  ogg">
  <source src="song.mp3" type="audio/
  mpeg">
Your browser does not support the audio
  tag.
</audio>
```

HTML5音频格式

当前，HTML5 Audio标签支持三种格式的音频，分别是wav mp3和ogg格式。而目前主流浏览器对他们的支持如下：

	IE 9	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
Ogg Vorbis		√	√	√	
MP3	√			√	√
Wav		√	√		√

IE8以及IE8以下不支持HTML5的audio标签

HTML 5 Audio属性

1.autoplay:唯一可选值为autoplay，出现autoplay属性并准确赋值时，音频将会自动播放

2.controls：唯一可选值为controls，出现controls属性并准确赋值时，音频播放控件将会显示，控件包括：播放、暂停、定位、音量、全屏切换、字幕（如果可用）、音轨（如果可用）。

3.loop：唯一可选值为loop，出现loop属性并准确赋值时，音频将会循环播放。

4.preload：可选值有auto(当页面加载后载入整个音频)、meta(当页面加载后只载入元数据)和none(当页面加载后不载入音频) 如果设置了前面的autoplay属性，那么preload将会被忽略。

5.src：指定音频URL地址，可以是相对的URL也可以是绝对的URL 当然还可以像第2和第3个例子一样，用source标签来指定源。

■

大数据与商业分析：新的TOP-DOWN纠结

■ 本文将给大家介绍大数据时代的商业分析模式，期间有作者本人的疑惑与深思，供大家参考。

---------回顾早期的争论----------

现在搞大数据（后端数据）的朋友，和早些年搞数据仓库的没啥本质区别，因为都是人为需要把各系统的数据集中化，现在增加个非结构化数据，于是就叫大数据，而以前就叫数据仓库。

在10多年前，数据仓库界一直有个争论，到底是TOP-DOWN好还是Bottom-UP好，一边说，我先把企业所有数据都集成，数据基础做扎实了，然后分析就可以源源不断产出了。而另一边认为，我们得先建数据集市，把业务主题搞扎实，让数据先有产出，然后再多个主题后再进行数据梳理整合。

前者的诟病是大半年都不会有啥“数据变现”的产出，问产出，就一直在说，系统搭建中，XX系统搭好了，XX业务数据集成了，等等。后者的诟病是，数据是有产出了，但是不同主题间的数据会有管理问题、每个主题需要不断添加新的数据源，开发、数据冗余都是个问题。

这是数据仓库界争论10多年的话题，直到近两年大数据兴起，大家就都没兴趣讨论了。在很多企业，大家都采用数据团队组织基础数据，做传统数据仓库前者的事情，再组织一个商业数据分析团队，做后者的事情。只不过大家都很少提数据仓库、BI这些老概念，不知道认为过时了，还是新一代数据人都不知道这些历史。其实本质是一样的，IBM等企业10年前就是2个团队在做，前后者的优势兼得。

--------商业数据分析的传统----------

在商业数据分析方面，我也是近4、5年才真正切入，之前都是做传统的数据仓库和BI。不过我经常欣赏几十年前的小数据商业分析的成功案例，试想小数据时代商业分析如此成功，大数据时代炒得那么热，却炒来炒去，就那么几个案例，有意思么？

我这里还是介绍那2个经典小数据时代的案例：

1.早期超市为了优化超市设计，采用人工观察，到后来的视频观察，来解析用户行为，他们解析用户行为的时候，发现了几个方面的业务改进Idea，超出了预期的想法。当看到人们眼光主要放在眼睛上下30度范围，于是陈列的时候，总是把利润最高的放这2排，利润低，性价比高的分别放上面和最下面；当发现人们通过买面包顺便买红酒，而不是买红酒顺便买面包，于是他们改变两种商品的组合和区域顺序；当发现免费品尝某水果，可以带动新鲜水果以及高利润的纯果汁的时候，超市又改变了策略。

2.有个商场，纠结于传统方案是现代方案，于是两个方案同时上，一个是用咨询团队，一个是监控方案，数人数看人流。结果咨询团队的方案胜，原因是咨询团队虽然是小数据，但是他们的抽样数据都是按照经验选择恰当的样本，其次小数据的信息非常全面，包括人的性别、年龄、身份背景、收入水平（经验判断）、去哪个区域，买东西走到哪一步了（咨询、试穿、有没讨价还价），这些数据监

控都没法得出，所以监控数据全面，但无法给出任何实质价值意义的建议，都是些空话。

---大数据商业分析应该TO-DWON还是Botton-Up?-----

一种思路是需要足够大、全面的数据，没有解决不了的分析，一上来把所有可能的维度、数据分布、趋势都尝试一遍，认为没有规律也能看出规律了；一种是看问题，需要解决什么问题，就做什么分析，缺什么数据再从大数据中去取。

有人问，如果采用第二种方案，那么大数据价值何在，那么多数据都没利用起来？我想问，如果你想挖前面一座金山，你是挖一部分，然后淘一部分出来卖，还是把金山都挖完，然后一点点淘出来卖呢？当然你选择第一种，但你选择这个，你不会说浪费了整个金山，因为你知道你需一点点变现后才会有动力挖后面的。但那么多搞大数据的人不是这个思路呢？

看到这里，可能大家认为我偏向第二种方案，其实不全是，我一直认为任何方案都有其优势，中庸取其精华最好。第二种方案的死穴是，如果你问题没搞得透彻明白，那么分析的都是无用功，都是在错误的轨道上转圈。

-----什么是大数据时代好的分析方案？----

我们来看第一个超市案例，他通过一个简单的、人工判断过的“用户行为”数据解决了这么多各种业务难题（陈列、布局、商品搭配、促销等），是因为他们发现一个问题，然后去解决的思路么？不是，而是充分以“商业”为中心，以“迎合用户需求，满足商业利益最大化”这一个最大分析目的，发现了用户行为，原来对这么多商业布局都有改进的地方，可以同时满足用户的购物需求，也满足商业利益最大化的需求，于是产生了后来的一系列超市零售改进。

再看大数据的案例，某人收到小孩用品推荐，他很生气投诉，结果没多久就写感谢信，说他女儿真的怀孕了，感谢云云。其实从这个案例中，除非给人惊奇外，有多大“商业价值”？如果纯从数据商业价值来说，这个分析推荐的作用，比起超市零售通过用户行为的变革，简直不值得一提，为啥还广为流传？因为大数据时代，还找不出像样的案例，于是就拿惊奇的案例来吸引大众注意吧。

反过来说，并非大数据无用，或者说大数据不是远不够大（要知道比超市人工观察的数据大多了），而是大数据分析和应用的人们思维并未完全打开，我相信做推荐的朋友没多少熟悉供应链的，也没多少熟悉商品定价和生命周期管理的，所以他们的大数据只是应用在推荐。

结尾：话不能说满了，剩余的给大家来讨论、思考吧。■

